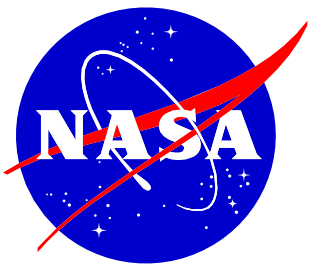
U.S. Department of the Interior
U.S. Geological Survey

LDCM CAL/VAL
ALGORITHM DESCRIPTION DOCUMENT

February 25, 2013
Version 3.0



Section 1 Document History

Document Number	Document Version	Publication Date	Change Number
LDCM-ADEF-001	Version 1.0	February 19, 2010	CCR 1102
LDCM-ADEF-001	Version 2.0	April 20, 2012	CCR 1341
LDCM-ADEF-001	Version 3.0	February 25, 2013	CCR 1419

Section 2 Introduction

The Landsat Data Continuity Mission's (LDCM), Landsat 8 (L8), is the latest satellite in the 40 year history of the Landsat program. Before the data are made available, they will be radiometrically and geometrically corrected using processing inputs from the Calibration Parameter File (CPF), and Bias Parameter File (BPF), and Radiometric Look-Up Table (RLUT). The Calibration Validation Team (CVT) will ensure that these files are monitored and updated over the life of the mission. The Image Assessment System (IAS) was developed to assess data on-orbit and to monitor changes temporally. The radiometric, geometric and spatial performance of the OLI and TIRS sensors will be continually monitored, characterized and calibrated on-orbit. Data that are processed by the LPGS system will also be trended to a database for later analysis by the Calibration Validation Team (CVT). The CVT will monitor the performance of L8 data on a daily basis by trending the results of radiometric and geometric algorithms processed on all data. Through regular evaluation of the stored results in the database, changes in instrument behavior can be monitored and corrected over time. The CVT will monitor the changes in the sensor and determine what should be updated in the CPF, BPF, and RLUT in order to create better image products while maintaining a level of consistency for comparability through time. This document details all of the radiometric and geometric processing algorithms for the image assessment and data processing of the Landsat 8 sensors.

Section 3 Document Overview

This document explains the methods for the geometric and radiometric characterization and calibration of the LDCM OLI and TIRS instruments implemented within the USGS EROS DPAS. A brief overview of the instruments and their data is provided, followed by discussions of the design philosophy, data flow diagrams and algorithm descriptions developed within the Cal/Val Toolkit (CVTK) for geometric and radiometric characterization and calibration.

Section 4 Instrument Overviews

The Landsat Data Continuity Mission (LDCM) is a joint mission formulated by the National Aeronautics and Space Administration (NASA) and the U.S. Geological Survey (USGS). The LDCM is a remote sensing satellite mission providing coverage of the Earth's land surfaces. This mission continues the 30+ years of global data collection and distribution provided by the Landsat series of satellites.

The space segment consists of an observatory that will be launched into a 705 km, 10:00 AM equatorial crossing sun synchronous orbit consistent with Landsat-7. The spacecraft will accommodate the OLI and TIRS. The spacecraft is being developed by General Dynamics Aerospace Information Systems.

4.1 OLI

The OLI instrument will image the Earth in 9 spectral bands which cover the visible, near-Infrared (VNIR) and Short Wave IR (SWIR) portions of the electromagnetic spectrum (see Table 1). Seven of the spectral bands are narrowed and refined from the Landsat-7 Enhanced Thematic Mapper Plus (ETM+) bands; a coastal/aerosol and a cirrus detection band have been added. All bands will be acquired at 12-bit radiometric resolution; 8 bands will be 30 meters and 1 band, the panchromatic band, will be 15 meters (see Table 1).

The OLI instrument is a pushbroom sensor being supplied by Ball Aerospace Technology Company. The telescope contains four mirrors with a front aperture stop that is 135 mm. The Focal Plane Array (FPA) is comprised of 14 Sensor Chip Assemblies (SCA) as shown in Figure 1 that is passively cooled. Each SCA contains 494 detectors with an additional 12 video reference pixels that don't respond to light.

4.1.1 On-Board Calibrators

The OLI provides both internal calibration sources such as lamps to ensure radiometric accuracy as well as capabilities to perform solar and lunar calibrations within the field of view constraints.

Solar Calibration and Linearity: The spacecraft must point the sun-viewing boresight at the sun and track it. To assure the calibration returns valid results, there must be a glint-free field of view for the diffuser as defined in the ICD. During solar looks, the solar array will be angled to prevent it from infringing on the glint-free field of view.

Lunar Calibration: The spacecraft must perform sweeps across the moon to image the moon on all 14 FPMs. Since the moon is only large enough to subtend on 1 FPM, it will require 14 sweeps across the moon (over multiple orbits if necessary) with the spacecraft yawing to place the moon on each of the FPMs.

4.1.2 Geolocation Calibration Activities

The following geolocation calibration activities require spacecraft operations:

Star Field Calibration: A section of the sky that is visible to both star trackers and the instrument must be imaged to locate stars against the star catalog. This will occur during commissioning only.

Calibration Data: In order to perform the calibration, the raw star tracker and gyro data for ground processing must be available.

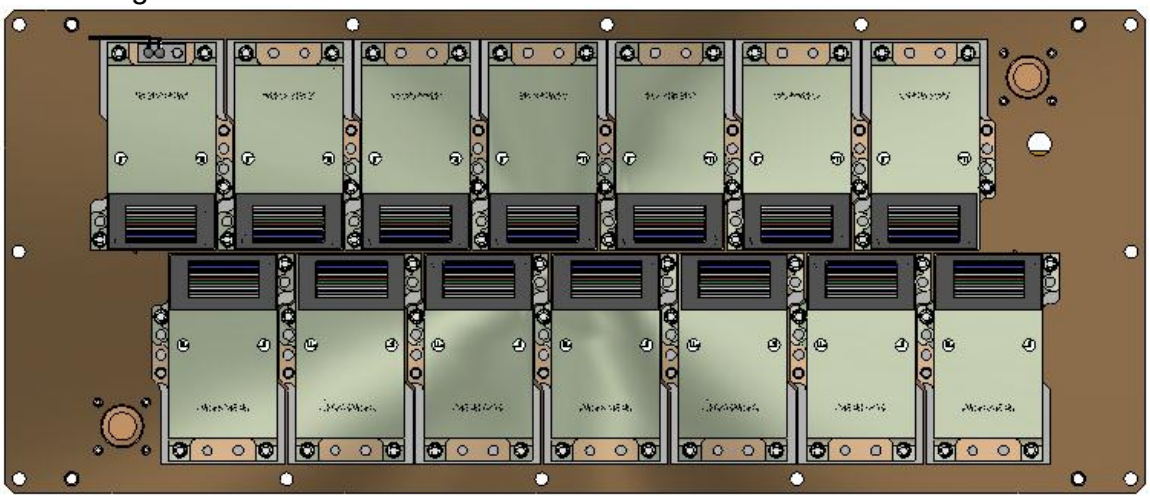


Figure 1: OLI focal plane assembly

Table 1: Spectral ranges and pixel sizes of OLI bands

#	Band Center	Wavelength (nm)	Center Wavelength Tolerance (±nm)	Minimum Lower Band Edge (nm)	Maximum Upper Band Edge (nm)
1	Coastal Aerosol	443	2	433	453
2	Blue	482	5	450	515
3	Green	562	5	525	600
4	Red	655	5	630	680
5	NIR	865	5	845	885
6	SWIR1	1610	10	1560	1660
7	SWIR2*	2200	10	2100	2300
8	Panchromatic**	590	10	500	680
9	Cirrus	1375	5	1360	1390

4.2 TIRS

The Thermal Infrared Sensor (TIRS) is a 2-band thermal imager at 10.8 and 12 microns. Both bands will have a spatial resolution of 100 meters operating in a pushbroom method to achieve a 188-km swath width. The Focal Plane Array (FPA) is comprised of 3 Sensor Chip Assemblies with Quantum Well Infrared Photometers (QWIPs), and will be built in-house at the NASA Goddard Space Flight Center. The FPA will be cryo-cooled to 43 K with an optical assembly passively cooled to 180K. A scene select mirror in the optical path will allow calibration with 2 sources; a variable temperature blackbody and space views.

The optical design is a four-element refractive design with a 107.8 mm clear aperture. Three of the elements are based on germanium and the fourth on zinc selenide. TIRS has two spectral bands achieved through interference filters. The filters are thermally connected to the focal plane and operate at somewhat higher temperature. Transmission characteristics are tailored for each band.

Very good out of band rejection is required to perform precise spectral radiometry and the in-band transmission must be high enough to meet the detector sensitivity goals. In addition, filter placement must accommodate a 2.5 second simultaneity requirement between 10.8 and 12 um measurements and all data must be collected within 170 rows of detector pixels.

TIRS relies on QWIP detectors coupled with existing Indigo 9803 640 x 512 pixel ROICs to give the previously-mentioned 185 km swath in 3 arrays with 35 pixel overlap between arrays.

4.2.1 Onboard calibrator

A key component for the TIRS sensor is the onboard calibrator. The calibrator will be a curved-plate blackbody with V-grooves to improve emissivity. The design and coating will be very similar to that used for MODIS to give high emissivity and controllable temperature. The output from the blackbody will be NIST traceable and capable of providing sources of two temperatures between 265 and 330 K within two orbits. Set point control of the blackbody will be 2 K with the capability to change the temperature by 6 K per half orbit.

4.2.2 Scene-select mirror

A scene-select mirror rotates around the optical axis on a 45-degree plane to provide the telescope with a view to nadir (earth), space (cold calibration “target”), and on-board blackbody (hot calibration target). The mirror is based on a solid aluminum blank diamond turned flat and super polished. The size of the mirror is 206.5 x 148.5 mm

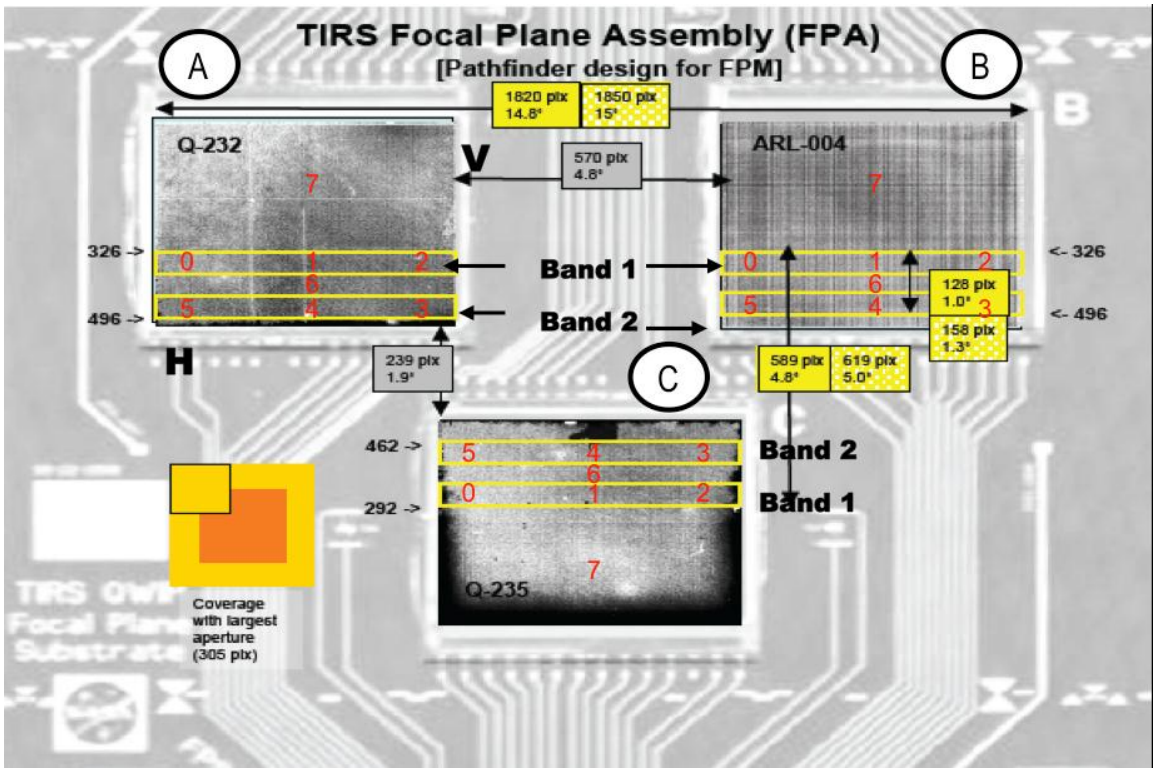


Figure 2: TIRS focal plane assembly

Table 2: Spectral ranges and pixel sizes of TIRS bands

#	Band	Center Wavelength (nm)	Center Wavelength Tolerance (±nm)	Minimum Lower Band Edge (nm)	Maximum Upper Band Edge (nm)
10	Thermal 1	10800	200	10300	11300
11	Thermal 2	12000	200	11500	12500

Section 5 Characterization and Calibration Overview

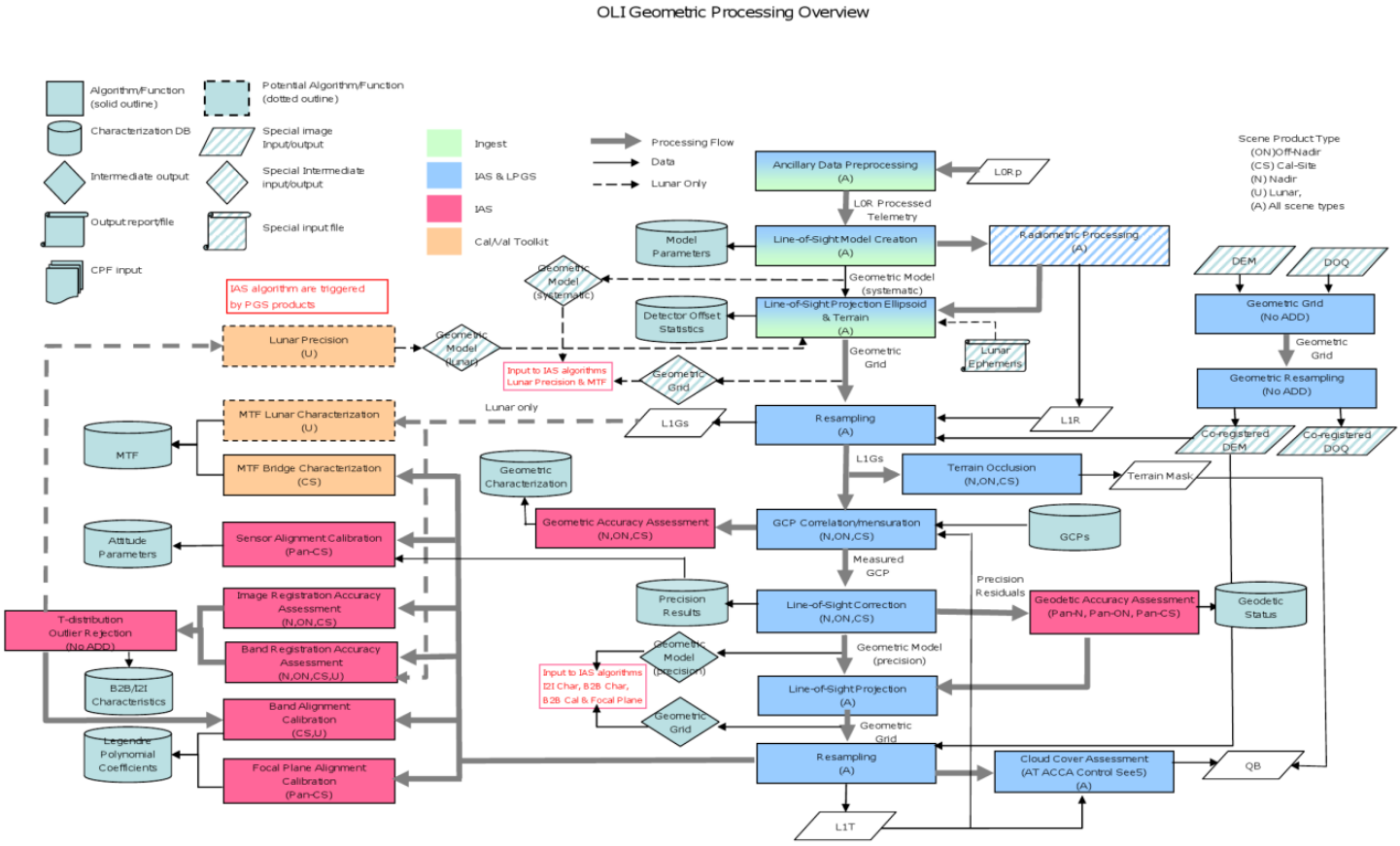
As used in this document, characterization is the process of measuring and evaluating the geometric and radiometric performance of the OLI and TIRS instruments. Calibration is the process of using the information obtained during characterization to update the calibration parameters associated with the both instruments. The following characterization and calibration routines for each instrument are discussed.

All calibration parameters are stored in an American Standard Code for Information Interchange (ASCII) file that can be accessed during processing. The file containing these parameters is referred to as the Calibration Parameter File (CPF) in this document.

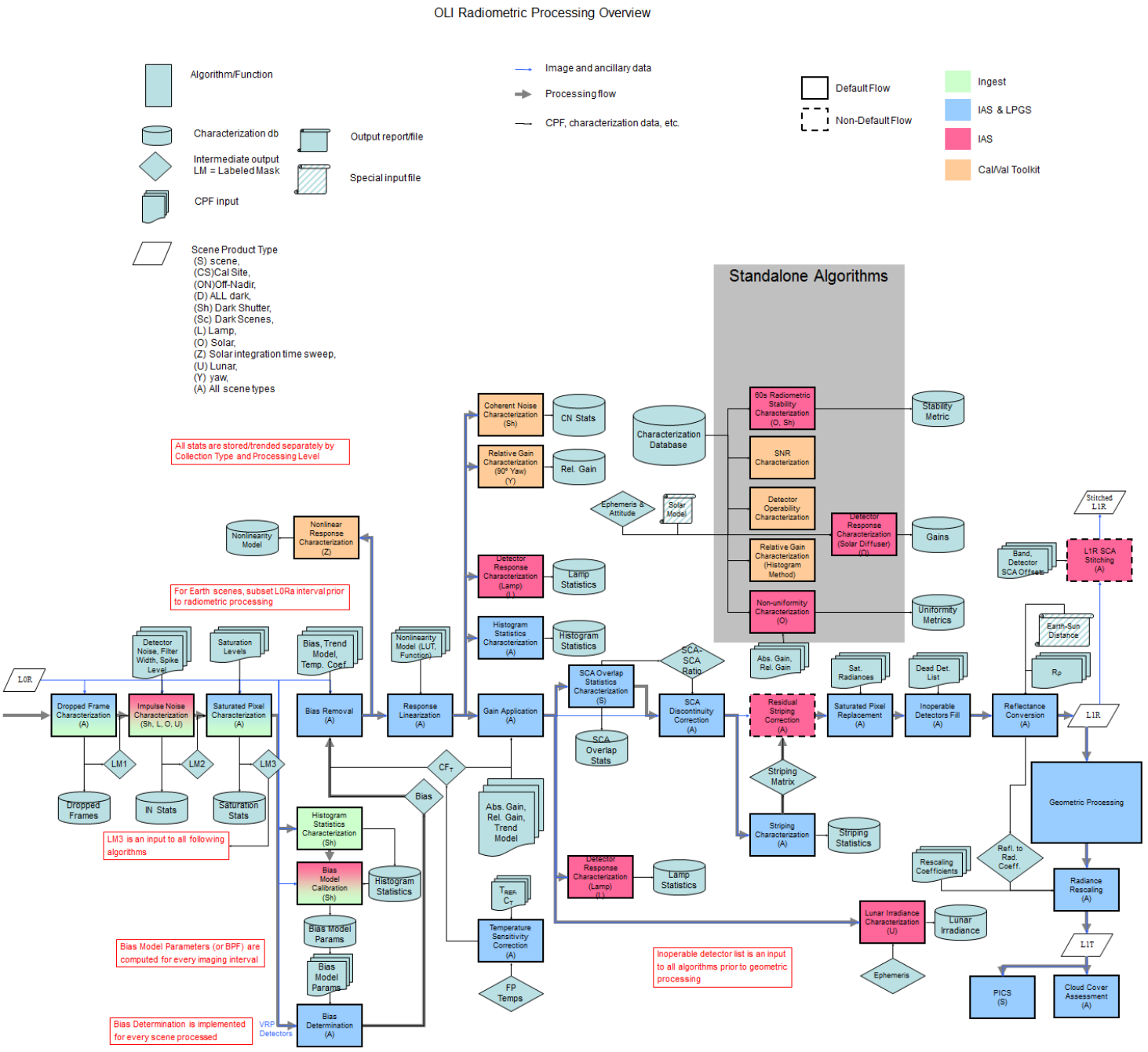
Section 6 Process Flows

In order to perform the characterization, calibration and correction functions, described in the LDCM Government Calibration and Validation Plan, each of the algorithms (described below in Sections 3 and 4) are linked together in a processing flow. This processing flow is separated into OLI geometry, OLI radiometry, TIRS geometry, and TIRS radiometry.

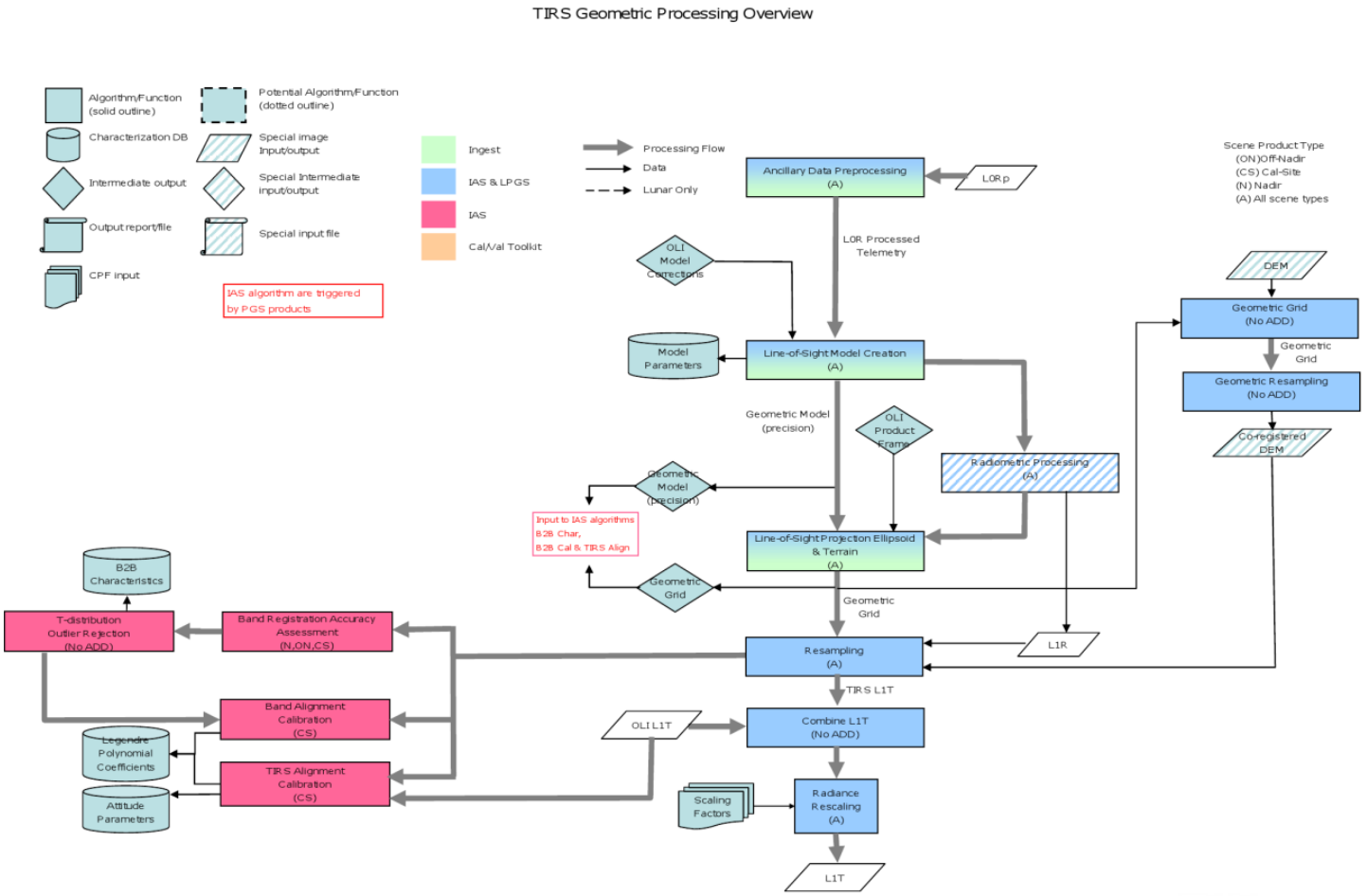
6.1 OLI Geometry



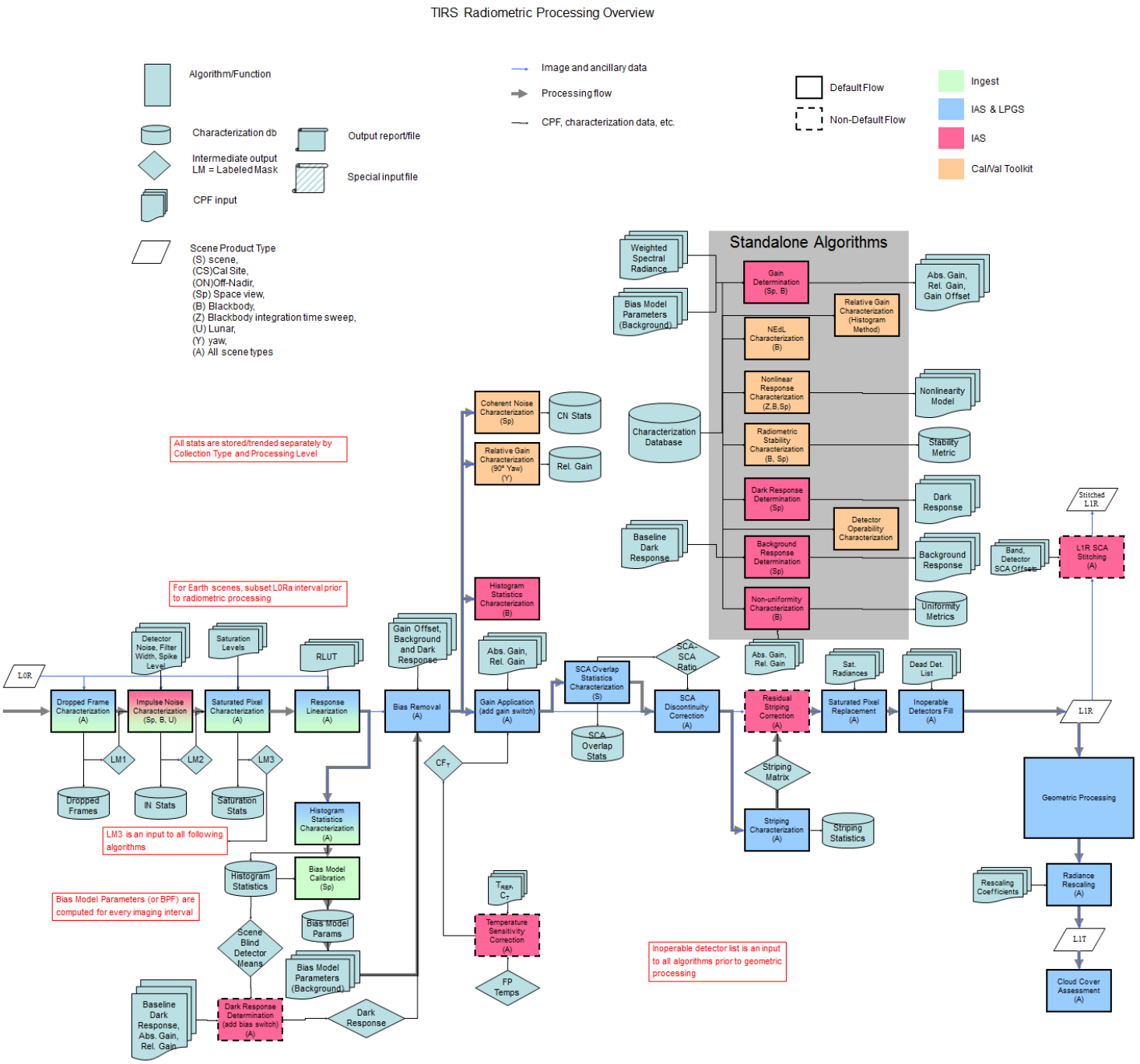
6.2 OLI Radiometry



6.3 TIRS Geometry



6.4 TIRS Radiometry



Section 7 Algorithms

7.1 Common Geometry Algorithms

7.1.1 Coordinate Systems

7.1.1.1 Coordinate System Definitions

There are ten coordinate systems used by the LDCM IAS geometric algorithms. These coordinate systems are referred to frequently in the remainder of this document and are briefly defined here to provide context for the subsequent discussion. They are presented in the order in which they would be used to transform a detector and sample time into a ground position.

1. OLI Instrument Line-of-Sight (LOS) Coordinate System

   The OLI LOS coordinate system is used to define the band and detector pointing directions relative to the instrument axes. These pointing directions are used to construct LOS vectors for individual detector samples. This coordinate system is defined so that the Z-axis is parallel to the telescope boresight axis and is positive toward the OLI aperture. The origin is where this axis intersects the OLI focal plane. The X-axis is parallel to the along-track direction, with the positive direction toward the leading, odd numbered, SCAs (see Figure 6.1.1.1-1). The Y-axis is in the across-track direction with the positive direction toward SCA01. This definition makes the OLI coordinate system nominally parallel to the spacecraft coordinate system, with the difference being due to residual misalignment between the OLI and the spacecraft body.

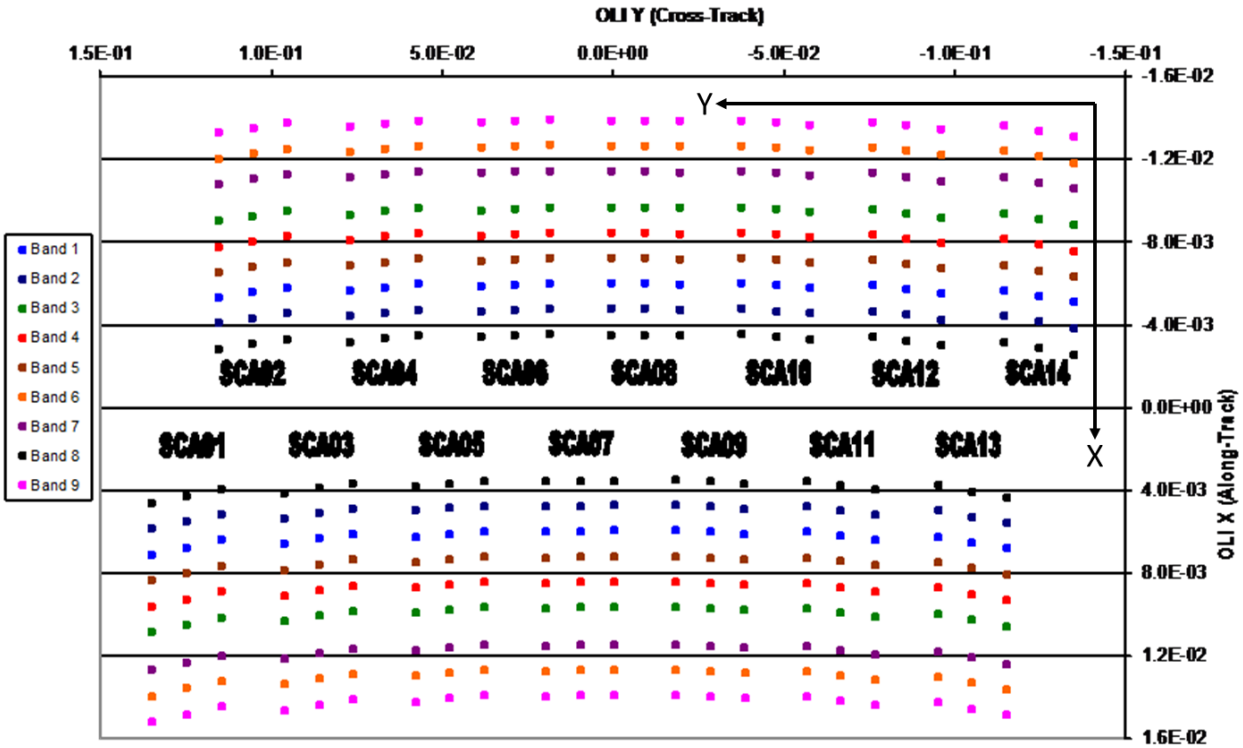


Figure 6.1.1.1.1. OLI Line-of-Sight Coordinate System

2. TIRS Instrument Coordinate System

The orientations of the TIRS detector LOS directions and of the TIRS scene select mirror (SSM) are both defined within the TIRS instrument coordinate system. TIRS LOS coordinates define the band and detector pointing directions relative to the instrument axes. These pointing directions are used to construct LOS vectors for individual detector samples. These vectors are reflected off of the SSM to direct them out the TIRS aperture for Earth viewing. The TIRS LOS model is formulated so that the effect of a nominally pointed SSM is included in the definition of the detector lines-of-sight, with departures from nominal SSM pointing causing perturbations to these lines-of-sight. This formulation allows TIRS LOS construction to be very similar to OLI, and is described in detail below, in the TIRS Line-of-Sight Model Creation algorithm.

The TIRS coordinate system is defined so that the Z-axis is parallel to the TIRS boresight axis and is positive toward the TIRS aperture. The origin is where this axis intersects the TIRS focal plane. The X-axis is parallel to the along-track direction, with the positive direction toward the leading SCA, SCA02 (see Figure 6.1.1.1-2). The Y-axis is in the across-track direction with the positive direction toward SCA03. This definition makes the TIRS coordinate system nominally parallel to the spacecraft coordinate system, with the difference being due to residual misalignment between the TIRS and the spacecraft body.

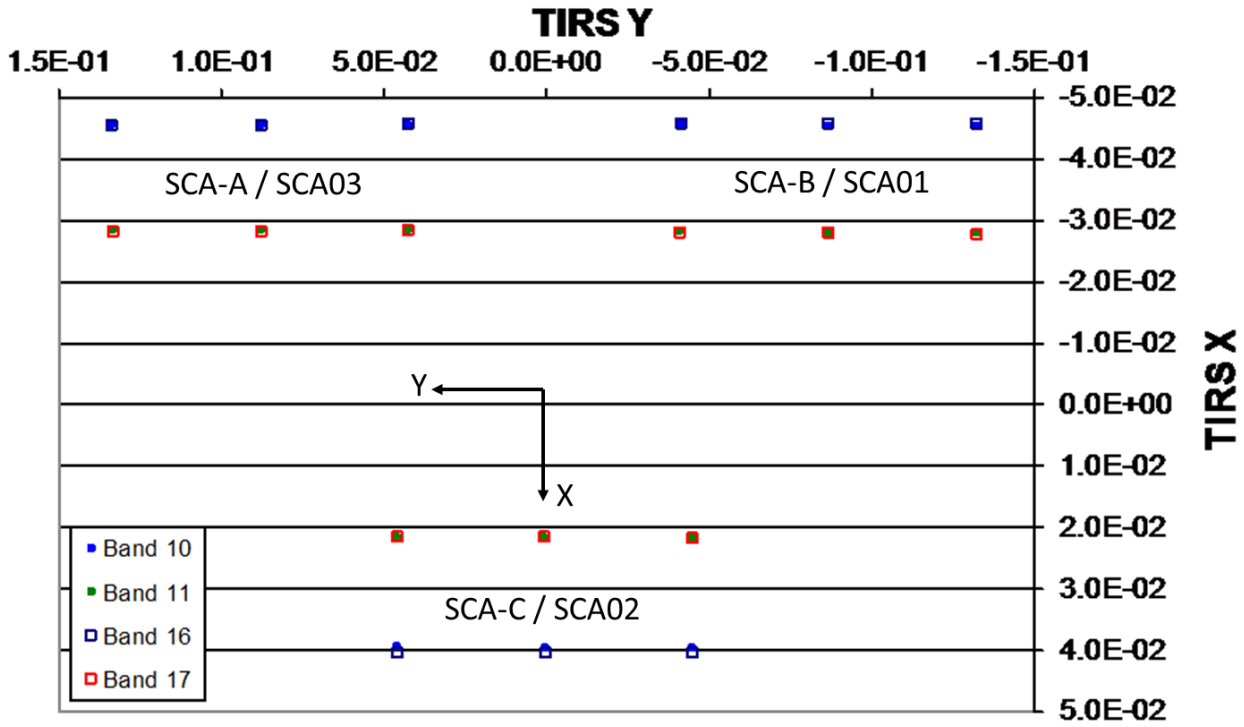


Figure 6.1.1.1-2. TIRS Line-of-Sight Coordinates

3. Spacecraft Coordinate System

The spacecraft coordinate system is the spacecraft-body-fixed coordinate system used to relate the locations and orientations of the various spacecraft components to one another and to the OLI and TIRS instruments. It is defined with the +Z axis in the Earth facing direction, the +X axis in the nominal direction of flight, and the +Y axis toward the cold side of the spacecraft (opposite the solar array). This coordinate system is useful during observatory integration and prelaunch test where it is used to determine the prelaunch positions and alignments of the

attitude control sensors (star trackers and SIRU) and instrument payloads (OLI and TIRS). The spacecraft coordinate system is nominally the same as the navigation reference system (see below) used for spacecraft attitude determination and control. However, for reasons explained below, these two coordinate systems are treated separately.

4. Navigation Reference Coordinate System

   The navigation reference frame (a.k.a., the attitude control system reference) is the spacecraft-body-fixed coordinate system used for spacecraft attitude determination and control. The coordinate axes are defined by the spacecraft attitude control system (ACS), which attempts to keep the navigation reference frame aligned with the (yaw-steered) orbital coordinate system (for nominal nadir pointing) so that the OLI and TIRS boresight axes are always pointing toward the center of the Earth. It is the orientation of this coordinate system relative to the inertial coordinate system that is captured in spacecraft attitude data.

   Ideally, the navigation reference frame is the same as the spacecraft coordinate system. In practice, the navigation frame is based on the orientation of the absolute attitude sensor (i.e., star tracker) being used for attitude determination. Any errors in the orientation knowledge for this tracker with respect to the spacecraft body frame will lead to differences between the spacecraft and navigation coordinate systems. This becomes important if the absolute attitude sensor is changed, for example by switching from the primary to the redundant star tracker during on-orbit operations. Such an event would effectively redefine the navigation frame to be based on the redundant tracker with the difference between the spacecraft and navigation frames now resulting from redundant tracker alignment knowledge errors, rather than from primary tracker alignment knowledge errors. This redefinition would require updates to the on-orbit instrument-to-ACS alignment calibrations. So, the spacecraft and navigation reference coordinate systems are different because the spacecraft coordinate system is fixed but the navigation reference can change.

5. SIRU Coordinate System

   The spacecraft orientation rate data provided by the spacecraft attitude control system's inertial measurement unit are referenced to the Space Inertial Reference Unit (SIRU) coordinate system. The SIRU consists of four rotation-sensitive axes. This configuration provides redundancy to protect against the failure of any one axis. The four SIRU axis directions are determined relative to the SIRU coordinate system, the orientation of which is itself measured relative to the spacecraft coordinate system both prelaunch and on-orbit, as part of the ACS calibration procedure. This alignment transformation is used by the IAS to convert the SIRU data contained in the LDCM spacecraft ancillary data to the navigation reference coordinate system for blending with the ACS quaternions.

6. Orbital Coordinate System

   The orbital coordinate system is centered at the spacecraft, and its orientation is based on the spacecraft position in inertial space (see Figure 6.1.1.1-3). The origin is the spacecraft's center of mass, with the Z-axis pointing from the spacecraft's center of mass to the Earth's center of mass. The Y-axis is the normalized cross product of the Z-axis and the instantaneous (inertial) velocity vector and corresponds to the negative of the instantaneous angular momentum vector direction. The X-axis is the cross product of the Y and Z-axes. The

orbital coordinate system is used to convert spacecraft attitude, expressed as ECI quaternions, to roll-pitch-yaw Euler angles.

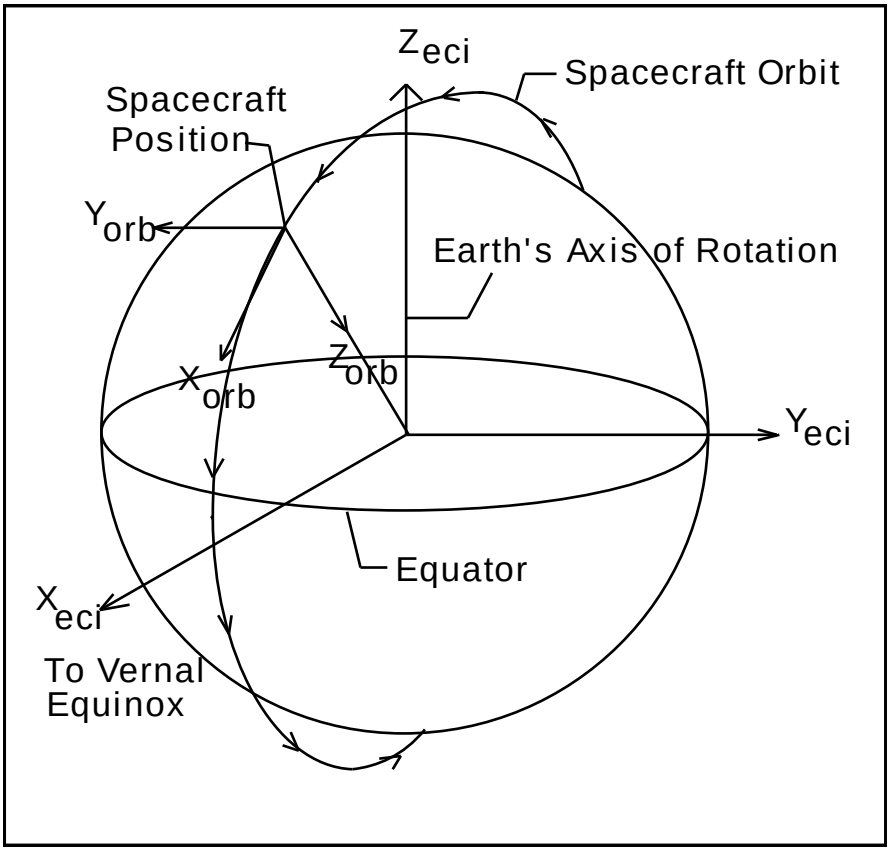


Figure 6.1.1.1-3. Orbital Coordinate System

7. ECI J2000 Coordinate System

   The Earth-Centered Inertial (ECI) coordinate system of epoch J2000 is space-fixed with its origin at the Earth's center of mass (see Figure 6.1.1.1-). The Z-axis corresponds to the mean north celestial pole of epoch J2000.0. The X-axis is based on the mean vernal equinox of epoch J2000.0. The Y-axis is the cross product of the Z and X axes. This coordinate system is described in detail in the Explanatory Supplement to the Astronomical Almanac published by the U.S. Naval Observatory. Data in the ECI coordinate system are present in the LDCM spacecraft ancillary data form of attitude quaternions that relate the navigation frame to the ECI J2000 coordinate system.

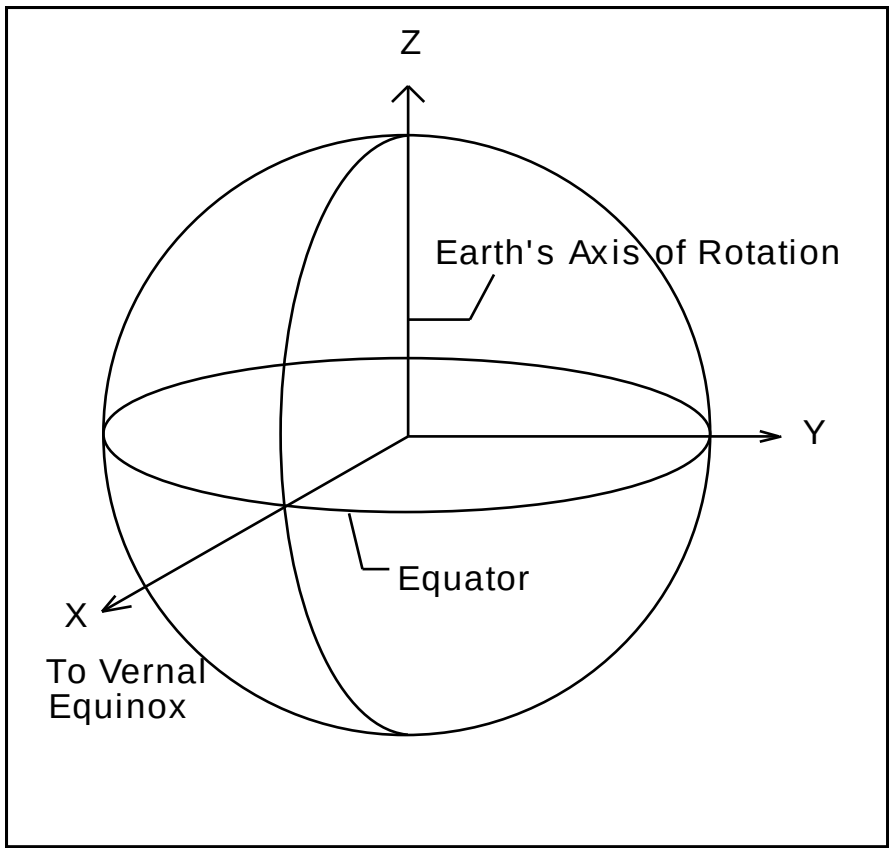


Figure 6.1.1.1-4. Earth-Centered Inertial (ECI) Coordinate System

8. ECEF Coordinate System

   The Earth-Centered Earth Fixed (ECEF) coordinate system is Earth-fixed with its origin at the Earth's center of mass (see Figure 6.1.1.1-). It corresponds to the Conventional Terrestrial System defined by the Bureau International de l'Heure (BIH), which is the same as the U.S. Department of Defense World Geodetic System 1984 (WGS84) geocentric reference system. This coordinate system is described in the Supplement to Department of Defense World Geodetic System 1984 Technical Report, Part 1: Methods, Techniques, and Data Used in WGS84 Development, TR 8350.2-A, published by the National Geospatial-Intelligence Agency (NGA)**Error! Reference source not found.**.

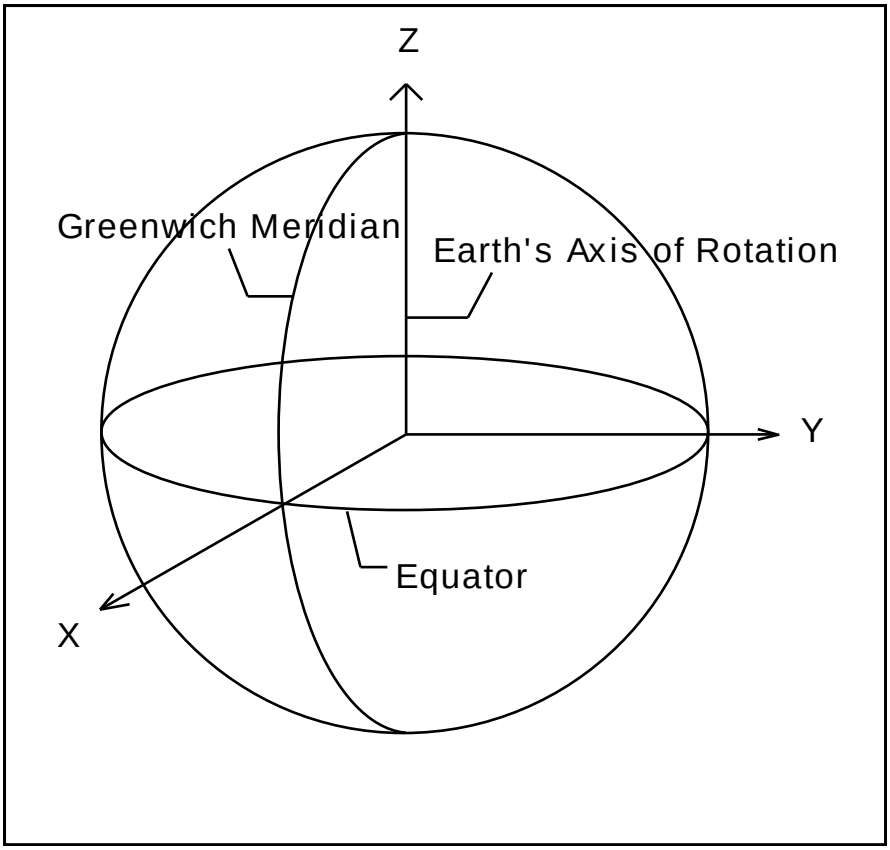


Figure 6.1.1.1-5. Earth-Centered Earth Fixed (ECEF) Coordinate System

9. Geodetic Coordinate System

   The geodetic coordinate system is based on the WGS84 reference frame with coordinates expressed in latitude, longitude, and height above the reference Earth ellipsoid (see Figure 6.1.1.1-). No ellipsoid is required by the definition of the ECEF coordinate system, but the geodetic coordinate system depends on the selection of an Earth ellipsoid. Latitude and longitude are defined as the angle between the ellipsoid normal and its projection onto the equator and the angle between the local meridian and the Greenwich meridian, respectively. The scene center and scene corner coordinates in the Level 0R product metadata are expressed in the geodetic coordinate system.

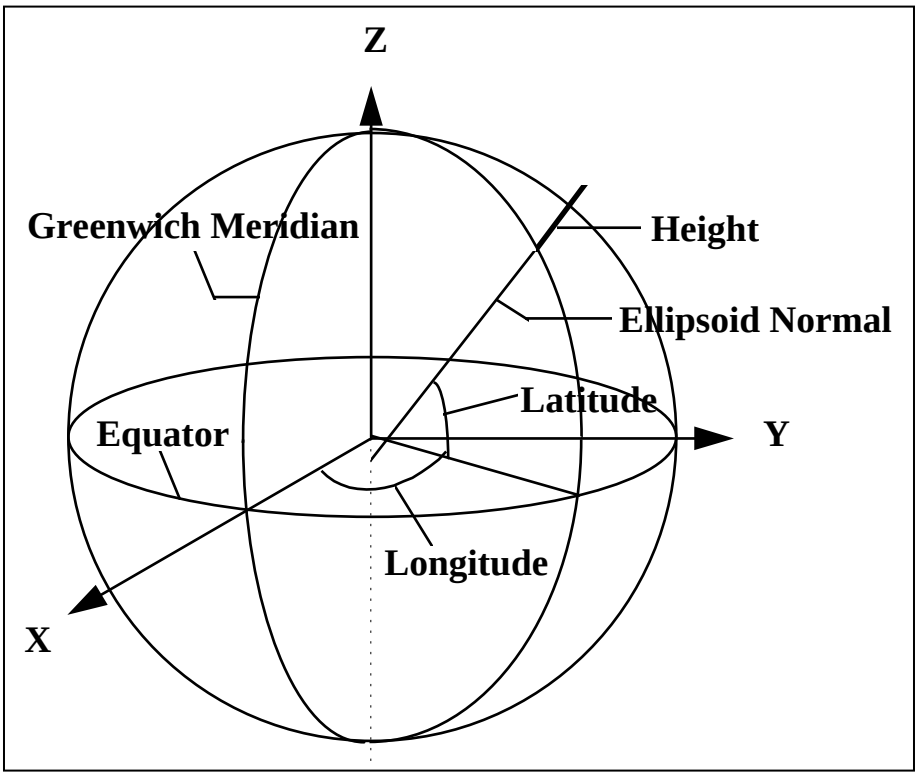


Figure 6.1.1.1-6. Geodetic Coordinate System

10. Map Projection Coordinate System

    Level 1 products are generated with respect to a map projection coordinate system, such as the Universal Transverse Mercator, which provides mapping from latitude and longitude to a plane coordinate system that is an approximation of a Cartesian coordinate system for a portion of the Earth's surface. It is used for convenience as a method of providing digital image data in an Earth-referenced grid that is compatible with other ground-referenced data sets. Although the map projection coordinate system is only an approximation of a true local Cartesian coordinate system at the Earth's surface, the mathematical relationship between the map projection and geodetic coordinate systems is defined precisely and unambiguously.

7.1.1.2 Coordinate Transformations

There are eight key transformations that relate the ten coordinate systems used by the IAS geometric algorithms. These transformations are referred to frequently in the remainder of this document and are defined here. They are presented in the logical order in which a detector and sample number would be transformed into a ground position.

1. OLI-to-Navigation Reference Transformation

The relationship between the OLI instrument and navigation reference coordinate systems is described by the OLI instrument alignment matrix. The transformation from sensor coordinates to navigation reference coordinates is a three-dimensional rotation, implemented as a matrix multiplication, and an offset to account for the distance between the ACS reference and the instrument aperture. This spacecraft center of mass-to-sensor offset is measured prelaunch and is not expected to be updated on-orbit. The ACS-to-OLI transformation matrix is initially defined as a static (non-time varying) rotation, with improved estimates provided post-launch. Subsequent analysis may detect repeatable variations with time, which can be effectively modeled, making this a (slowly) time-varying transformation. The nominal rotation matrix is the identity matrix.

2. TIRS-to-Navigation Reference Transformation

The relationship between the TIRS instrument and navigation reference coordinate systems is described by the TIRS instrument alignment matrix. Like the OLI, the transformation from sensor coordinates to navigation reference coordinates is a three-dimensional rotation, implemented as a matrix multiplication, and an offset to account for the distance between the ACS reference and the instrument aperture. This spacecraft center of mass-to-sensor offset is measured prelaunch and is not expected to be updated on-orbit. The ACS-to-TIRS transformation matrix measured directly prelaunch. Post-launch, improved estimates will be provided by estimating the OLI-to-TIRS alignment and combining that with the ACS-to-OLI alignment mentioned above. Note that any TIRS pointing offsets that are due to errors in SSM alignment knowledge will be attributed to the overall ACS-to-TIRS alignment by the on-orbit calibration. The nominal ACS-to-TIRS rotation matrix is the identity matrix.

3. SIRU-to-Navigation Reference Transformation

The SIRU coordinate system is related to the navigation reference coordinate system by the SIRU alignment matrix, which captures the orientation of the SIRU axes with respect to the navigation base. This transformation is applied to the SIRU measurements present in the spacecraft ancillary data prior to their integration with the ACS quaternions. The SIRU alignment is measured pre-flight and is nominally oriented with a 45-degree rotation about the X-axis, relative to the spacecraft/navigation reference coordinate system.

4. Navigation Reference-to-Orbital Transformation

The relationship between the navigation reference and orbital coordinate systems is defined by the spacecraft attitude. This transformation is a three-dimensional rotation matrix with the components of the rotation matrix being functions of the spacecraft roll, pitch, and yaw attitude angles. The nature of the functions of roll, pitch, and yaw depends on the exact definition of these angles. The conventions adopted in the LDCM model are described below in the Ancillary Data Preprocessing algorithm. Since the spacecraft attitude is constantly changing, this transformation is time varying. The nominal rotation matrix consists of a latitude-dependent rotation about the Z-axis (yaw). This “yaw-steering” is designed to compensate for the effects of Earth rotation as spacecraft motion passes the OLI and TIRS detector arrays over the Earth’s surface.

5. Orbital-to-ECI Transformation

The relationship between the orbital and ECI coordinate systems is based on the spacecraft's instantaneous ECI position and velocity vectors. The rotation matrix to convert from orbital to ECI can be constructed by forming the orbital coordinate system axes in ECI coordinates:

$\underline{P}$ = spacecraft position vector in ECI
$\underline{V}$ = spacecraft velocity vector in ECI
$T_{eci/orb}$ = rotation matrix from orbital to ECI

$\underline{b}_3 = -\underline{p} / |\underline{p}|$ (nadir vector direction)
$\underline{b}_2 = (\underline{b}_3 \times \underline{v}) / |\underline{b}_3 \times \underline{v}|$ (negative of angular momentum vector direction)
$\underline{b}_1 = \underline{b}_2 \times \underline{b}_3$ (circular velocity vector direction)
$T_{eci/orb} = [\ \underline{b}_1\ \ \underline{b}_2\ \ \underline{b}_3\ ]$

6. ECI-to-ECEF Transformation

The transformation from ECI-to-ECEF coordinates is a time-varying rotation due primarily to the Earth's rotation, but it also contains more slowly varying terms for precession, astronomic nutation, and polar wander. The ECI-to-ECEF rotation matrix can be expressed as a composite of these transformations:

$T_{ecr/eci} = A\ B\ C\ D$

A = polar motion
B = sidereal time
C = astronomic nutation
D = precession

Each of these transformation terms is described in more detail below in the Ancillary Data Preprocessing algorithm. Note that LDCM uses the precession, nutation, and sidereal time definitions from the IAU resolutions of 1997-2000 as described in U.S. Naval Observatory Circular 179. This is a newer formulation than was used in the heritage Landsat 7 system.

7. ECEF-to-Geodetic Transformation

The relationship between ECEF and geodetic coordinates can be expressed simply in its direct form:

$e^2 = 1 - b^2 / a^2$
$N = a / (1 - e^2 \sin^2(lat))^{1/2}$
$X = (N + h) \cos(lat) \cos(lon)$
$Y = (N + h) \cos(lat) \sin(lon)$
$Z = (N (1 - e^2) + h) \sin(lat)$

where:

X, Y, Z = ECEF coordinates
lat, *lon*, *h* = geodetic coordinates
N = ellipsoid radius of curvature in the prime vertical
e^2 = ellipsoid eccentricity squared

a, b = ellipsoid semi-major and semi-minor axes

The closed-form solution for the general inverse problem (the problem of interest here) involves the solution of a quadratic equation and is not typically used in practice. Instead, an iterative solution is used for latitude and height for points that do not lie on the ellipsoid surface.

8. Geodetic-to-Map Projection Transformation

The transformation from geodetic coordinates to the output map projection depends on the type of projection selected. The mathematics for the forward and inverse transformations for the Universal Transverse Mercator (UTM), Lambert Conformal Conic, Transverse Mercator, Oblique Mercator, Polyconic, Polar Stereo Graphic, and the Space Oblique Mercator (SOM) are given in John P. Snyder's Map Projections – A Working Manual, USGS Professional Paper 1395.

7.1.2 Time Systems

Four time systems are of primary interest for the IAS geometric algorithms: International Atomic Time (Temps Atomique International [TAI]), Universal Time—Coordinated (UTC), Universal Time—Corrected for polar motion (UT1), and Spacecraft Time (the readout of the spacecraft clock, derived from GPS time). Spacecraft Time is the time system used for the spacecraft time codes found in the Level 0R ancillary data (including image time codes). UTC is the standard reference for civil timekeeping. UTC is adjusted periodically by whole leap seconds to keep it within 0.9 seconds of UT1. UT1 is based on the actual rotation of the Earth and is needed to provide the transformation from stellar-referenced inertial coordinates (ECI) to terrestrial-referenced Earth-fixed coordinates (ECEF). TAI provides a uniform, continuous time stream that is not interrupted by leap seconds or other periodic adjustments. It provides a consistent reference for resolving ambiguities arising from the insertion of leap seconds into UTC, which can lead to consecutive seconds with the same UTC time. Spacecraft time is based on GPS time which is, itself, a fixed offset from TAI. These and a variety of other time systems, and their relationships, are described in the Explanatory Supplement to the Astronomical Almanac, mentioned previously. The significance of each of these time systems with respect to the IAS geometric algorithms is described below.

1. Spacecraft Time

In accordance with the LDCM Spacecraft to Ground Interface Control Document (70-P58230P, Rev C), the LDCM spacecraft clock reports time as TAI seconds since the spacecraft (J2000) epoch, defined as follows:

January 1, 2000, 11:59:27.816 TAI

which is the same as:

January 1, 2000, 11:58:55.816 UTC.

Epoch J2000 occurred at January 1, 2000 12:00:00 Barycentric Dynamical Time (TDB). At the time of the J2000 epoch, Terrestrial Dynamical Time (TDT) differed from TDB by approximately 73 microseconds (ref. Explanatory Supplement to the Astronomical Almanac).

This small difference is ignored in the definition above, and the epoch is effectively taken to be January 1, 2000, 12:00:00 TDT. Since TDT is defined to be TAI + 32.184 seconds, we have 11:59:27.816 TAI + 32.184 sec = 12:00:00 TDT. Furthermore, at the time of the J2000 epoch, TAI and UTC differed by 32 accumulated leap seconds, so 11:58:55.816 UTC + 32.000 sec = 11:59:27.816 TAI. Note from the above that the relationship between spacecraft time and TAI is fixed but the relationship between spacecraft time and UTC changes over time, with the offset increasing by one second each time a new leap second is declared. Note that as of the 2013 LDCM launch date, three additional leap seconds had been declared (in January 2006, January 2009, and July 2012).

The LDCM flight software maintains the accuracy of the spacecraft clock using time data from the on-board GPS receiver(s). The spacecraft clock is then used to time tag the spacecraft ancillary data and to provide a timing reference for the OLI and TIRS instruments. Spacecraft time is used to define the times at which the flight software generates filtered attitude and ephemeris estimates based on the input GPS, star tracker, and SIRU data. These estimates are included in the spacecraft ancillary data stream for use by ground processing. Also included in the ancillary data are the raw SIRU measurements. Individual SIRU observations are time tagged using a clock/counter internal to the SIRU itself, but the SIRU ancillary data also includes SIRU time synch events that make it possible to relate the SIRU clock to spacecraft time.

The spacecraft clock also provides time synchronization signals to the OLI and TIRS instruments once per second. Both instruments use this one pulse per second signal to regulate their internal clocks, thereby registering the image time codes to spacecraft time. Note that any instrument clock rate errors will be manifested as (small) step discontinuities in the image time codes, which correspond to the 1 PPS updates. The resulting time code irregularities are corrected when the OLI and TIRS geometric models are created, as described below in the OLI LOS Model Creation algorithm and the TIRS LOS Model Creation algorithm.

2. UTC

   As mentioned above, UTC is maintained within 0.9 seconds of UT1 by the occasional insertion of leap seconds. A table of leap seconds relating UTC to TAI is maintained in the LDCM Calibration Parameter File (CPF) to support the spacecraft time to UTC conversion. To convert spacecraft time to UTC, the number of additional leap seconds declared since the spacecraft epoch are subtracted from the reported spacecraft seconds since epoch and the result is added to the UTC representation of the epoch presented above. Leap second information can be obtained from the International Earth Rotation Service (IERS) in their Bulletin C publications.

3. UT1

   UT1 represents time with respect to the actual rotation of the Earth and is used by the IAS algorithms, which transform inertial ECI coordinates or lines of sight to Earth-fixed ECEF coordinates. Failure to account for the difference between UT1 and UTC in these algorithms can lead to ground position errors as large as 400 meters at the equator (assuming the maximum 0.9-second UT1-UTC difference). The UT1-UTC correction typically varies at the rate of approximately 2 milliseconds per day, corresponding to an Earth rotation error of about 1 meter. Thus, UT1-UTC corrections should be interpolated or predicted to the actual image

acquisition time to avoid introducing errors of this magnitude. The UT1-UTC offset, along with the polar wander Earth orientation parameters, can be obtained from IERS Bulletin B (for retrospective data) and Bulletin A (for predicted data). Tables of the UT1-UTC and polar wander Earth orientation parameters are also maintained in the LDCM CPF.

4. TAI

   Although the IAS algorithms do not operate directly in TAI, it underlies the definition of spacecraft time, as noted above. As such, it can be helpful to use TAI as a standard reference that can be related to UTC, using the CPF leap second file, and to spacecraft time, via the constant offset, to assist IAS operations staff in anomaly resolution.

7.1.3 Scene Framing Algorithm

7.1.3.1 Background/Introduction

The LDCM scene framing algorithm uses the spacecraft ancillary data, preprocessed to perform scaling, coordinate conversion and to repair errors, and the image timing information to determine the locations of scene centers within the interval. It then assigns Worldwide Reference System-2 (WRS-2) path row coordinates to these scene centers for purposes of subsequent metadata generation.

The Landsat heritage scene framing algorithm will be used to frame nadir-pointing data and to determine the nadir path/row references for off-nadir acquisitions, but the LDCM capability to point (roll) up to 15 degrees off-nadir will lead to data acquisitions that do not fall on the regular WRS-2 reference grid and will, in some cases, fall entirely outside the heritage WRS-2 coverage area for acquisitions near the poles. These additional complications require adjustments to the heritage algorithm to address both the scene definition (i.e., where do we declare the scene centers) aspect of scene framing and the WRS-2 grid (path/row) assignment aspect of scene framing. This algorithm addresses those requirements.

The algorithm developed here separates the scene definition and WRS-2 labeling aspects of the scene framing problem. It also uses different logic for high-latitude (polar region) and low latitude (non-polar) acquisitions. At high latitudes, off-nadir acquisitions will be poorly aligned with and will sometimes fall outside of the WRS-2 grid, so the heritage (nadir) orbit-based approach to scene center time definition is used in those areas. Using the nadir path/row for scene and interval identification also ensures unique ids as well as consistency with planned acquisitions. Furthermore, to help in identifying coverage of off-nadir acquisitions, especially near the poles, *target* WRS-2 labeling is determined and any imagery falling outside the WRS-2 grid uses special target row numbering.

For non-polar regions, the guiding principle is to make even off-nadir scenes as consistent as possible in coverage with the nadir acquisitions of the same region by using a "row-based" approach to scene definition. Defining scene centers at the locations where the Operational Land Imager (OLI)[1] boresight crosses the latitudes that correspond to WRS-2 row centers makes the off-nadir scene latitude bounds align with the nadir-viewing scenes from adjacent paths. This should lead to greater consistency in scene coverage and improve the interoperability of nadir and off-nadir data[2].

[1] For any Thermal Infrared Sensor (TIRS) only earth acquisitions, the TIRS boresight is used.
[2] In contrast, framing all off-nadir data based on the spacecraft position instead of the boresight location was rejected, as it would lead to off-nadir scenes exhibiting an along-track shift relative to the adjacent path nadir data.

Therefore, for non-polar regions, scene path/row computation is based on the boresight Line-of-Sight (LOS) intersection location. The basic principle in this portion of the algorithm is to treat the boresight location as if it were a sub-satellite point. We can then compute a corresponding orbital central travel angle and apparent descending node based on the nominal Landsat orbital inclination. The actual orbit data are used to determine whether the scene is ascending or descending mode and the central travel angle and descending node are adjusted accordingly. The central travel angle is used to derive the WRS-2 row and the descending node longitude yields the (fractional) path. Since the scene definition logic is defined by WRS-2 row crossings, for non-polar data the off-nadir scene paths will typically be fractional and the row will be an integer. Whereas in polar areas the off-nadir scenes target WRS-2 paths and rows will be fractional. These fractional WRS-2 path/rows will have to be rounded to the nearest path/row that represents the data. Target WRS-2 coordinates of nadir pointing scenes should be integers in both regions. Also, if the scene center falls too far off the WRS-2 grid, special target row numbers will be assigned (880-88x for the North Pole and 990-99x for the South Pole).

It's important to note that in the metadata the wrs_path/wrs_row values are always the nadir or orbital path/row of the satellite (and they're also used for the scene ids) and the *target* path/row is the LOS path/row. In the non-polar regions, the target row and orbital wrs_row should be equal for nadir viewing imagery. The target path should be between wrs_path-1 and wrs_path+1 at low latitudes but could vary by more at higher latitudes (approaching the polar regions). In the polar region, off-nadir target path/row will vary quite a bit from the orbital values.

7.1.3.2 Dependencies

The scene framing algorithm assumes that ancillary data for the full imaging interval with 8 seconds of extra ancillary data before and after the imagery, is available to provide the required geometric support data, that a Calibration Parameter File (CPF) containing Earth orientation parameters and OLI & TIRS alignment and offset information is available, and that the image time codes are available.

At a minimum four seconds of ancillary data are required before and after the imagery to construct consistent attitude and ephemeris time histories for the data set in order to achieve LDCM geolocation accuracy requirements. An additional four seconds is added as an allowance for late starts/stops, bad/partial frames at the beginning/end, etc… Also, not all of the instrument telemetry is actually updated in every ancillary data frame so it takes multiple (up to 4) frames to be guaranteed of getting a fresh sample.

For calibration collects (lamp, solar, shutter, black body, deep space, etc.) the geolocation framing is not done and a minimum of two seconds of ancillary are expected before and after the imagery.

A check to ensure the imagery is fully covered by ancillary data should be done at a minimum for all intervals. To avoid potential framing errors a check for at least 4 seconds of ancillary data before and after the imagery, for earth collects may be prudent. If there is insufficient ancillary data to cover the imagery or processing results in framing errors the imagery may have to be trimmed to fit within the available ancillary in order to process the data. A tool would have to be written to do this or ingest updated to "trim" imagery to fit within the ancillary data.

Addendum to image trimming: In order to support mission data files from a Live Downlink (like ICs would receive) image "trimming" is being looked at as an Ingest enhancement. Top option being considered is to <u>not</u> actually remove the imagery but to indicate an interval start/stop frame that fits

within the ancillary data, where processing would begin/finish. In that way no data is thrown away and there could potentially be updates to "extrapolate" the extent of the ancillary such that sometime in the future this imagery could be processed.

7.1.3.3 Inputs

The scene framing algorithm and its component sub-algorithms use the inputs listed in the following table. Note that some of these "inputs" are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs
ODL File (implementation)
CPF
Spacecraft TAI Epoch Time Reference
Earth orientation parameters (UT1UTC, pole wander, leap seconds)
OLI/TIRS Focal Plane Parameters
OLI/TIRS Parameters
Attitude Parameters
Orbit Parameters
WRS-2 Scene average elevation look-up file
Preprocessed Ancillary Data
Attitude Data
Attitude data UTC epoch: Year, Day of Year, Seconds of Day
Time from epoch (one per sample, nominally 50 Hz) in seconds
ECI quaternion (vector: q1, q2, q3, scalar: q4) (one per sample)
ECEF quaternion (one per sample)
Body rate estimate (roll, pitch, yaw rate) (one per sample) in radians/second
Roll, pitch, yaw estimate (one per sample) in radians
Ephemeris Data
Ephemeris data UTC epoch: Year, Day of Year, Seconds of Day
Time from epoch (one per sample, nominally 1 Hz) in seconds
ECI position estimate (X, Y, Z) (one set per sample) in meters
ECI velocity estimate (Vx, Vy, Vz) (one set per sample) in meters/second
ECEF position estimate (X, Y, Z) (one set per sample) in meters
ECEF velocity estimate (Vx, Vy, Vz) (one set per sample) in meters/second
L0R Data Contents
OLI Image Time Codes (one per frame)
TIRS Image Time Codes (one per frame)
Instrument-specific minimum number of frames per full scene (O:7001,T:2801)
Instrument-specific minimum number of overlap frames per scene (O:1322 [756 FOV + 566 overlap], T:1080 [800FOV + 168 overlap+ 112 misalignment])
Number of polar region rows for nadir scene center calculations (6)
Minimum latitude for special target row numbering (82.61deg) [Highest latitude if the roll shifts the imagery ~50% off the nadir pointing ground coverage.]

Note that both ECI and ECEF attitude and ephemeris data are specified as inputs but the baseline algorithm makes use of only the ECEF versions.

7.1.3.4 Outputs

The scene framing algorithm outputs are shown in the following table.

Scene Framing Data
Number of scenes in the interval
For each scene in the interval:
Scene center time: year, day of year, seconds of day UTC
Scene start time: year, day of year, seconds of day UTC
Scene stop time: year, day of year, seconds of day UTC
Scene corner information.
OLI start/center/stop frame numbers
TIRS start/center/stop frame numbers
Assigned orbital WRS-2 path
Assigned orbital WRS-2 row
Assigned target WRS-2 path
Assigned target WRS-2 row

7.1.3.4.1 Approach Overview

Scene Framing basically consists of the following:

a. Using the ephemeris data, determine the WRS-2 orbital (i.e. nadir) range – this is also called the "heritage method".
b. Determine scene center times for each row at the focal plane boresight position (OLI: Boresight is located between SCA7 & 8, TIRS: Boresight is located in the middle of SCA A, B, & C). This is performed by three different methods, depending on the row identified:
    i. LOS latitude crossing for non-polar rows,
    ii. Nadir position for Polar Region Rows (within +/-6 rows of the poles),
    iii. Zero Z-Velocity time for Polar Rows (246 or 122).
c. Determine each instrument's scene center frame number based on the center time, from which the start and stop frame numbers and times are also, determined using each instrument's minimum frame counts (given above).
d. Check and adjust scene start/stop times to ensure adequate overlap. This is mainly needed for off-nadir collects that transition to/from the polar region rows.
e. From each scene's center coordinates, determine the closest Target Path/Row. This helps determine closest WRS-2 coverage for off-nadir imaging. Also, for extremely high latitude, special target row numbers are used (88x and 99x) to help identify imagery viewed "off the WRS-2 grid" around the poles.
f. Scene metadata (corner points, sun azimuth/elevation, etc) are then completed.

7.1.3.5 Prototype Code

None

Note: The functionality described in this algorithm relies on geometric processing capabilities described in other LDCM/OLI/TIRS algorithm description documents. These algorithms will be referenced as necessary.

7.1.3.5.1 Scene Sizes

Full scene declaration is based on a minimum number of frames (OLI: 7001, TIRS:2801), anything less is considered a partial scene. See Appendix A: Scene Sizing Background for an overview of the

fields-of-view and calculations to define the number of OLI and TIRS frames in a full WRS-2 scene. The LDCM imaging instruments (OLI and TIRS) are push broom instruments with significantly large fields-of-view in the along-track direction. In addition, both instruments have redundant detectors which can be selected for active imaging. Since it is desirable to have a constant minimum frame size for Level-0 scenes to be considered "full", the minimum scene frame size for each instrument is set large enough to always ensure enough coverage to produce full Level-1 scenes which are approximately 180km in length. There's also a one second tolerance on the boresight alignment between the two instruments.

The standard length of a full Level-0 OLI scene is given by:

OLI Sample Rate	4.236	Milliseconds / frame
OLI FOV SCA Staggering	1.7	Degrees (along-track) The angle required for the leading and trailing imaging bands of the SCAs to cover a point on the ground relative to the center of the focal plane. See Appendix A
OLI FOV Frames	756	Frames (along-track)
WRS-2 Scene	24	Seconds (center-to-center)
	5664	Frames (center-to-center)
WRS-2 Scene Overlap	566	Frames (5% top & 5% bottom)
L0 Scene Length	6230	Frames (180km coverage, 5664+566)
	6986	Frames (adjusted for FOV, 6230+756)
	7001	15 frame cushion added (6986 + 15)
	29.67	Seconds for 7001 frames
Minimum OLI Overlap ($Start_i$-to-$Stop_{i-1}$)	1322	Frames (756 FOV + 566 overlap)

Similarly, the standard length of a full Level-0 TIRS scene is given by:

TIRS Sample Rate	14.286	Milliseconds / frame
TIRS FOV SCA Staggering	5.0	Degrees (along-track) The angle required for the leading and trailing imaging bands of the SCAs to cover a point on the ground relative to the center of the focal plane. See Appendix A
TIRS FOV Frames	800	Frames (along-track, includes science rows)
WRS-2 Scene	24	Seconds (center-to-center)
	1680	Frames (center-to-center)
WRS-2 Scene Overlap	168	Frames (5% top & 5% bottom)
L0 Scene Length	1848	Frames (180km coverage, 1680+168)
	2648	Frames (adjusted for FOV, 1848+800)
	2760	Frames (adjusted alignment tol, 2648+112)
	2801	41 frame cushion added (2760 + 41)
	40.01	Seconds for 2621 frames

Minimum TIRS Overlap ($Start_i$-to-$Stop_{i-1}$)	1080	Frames (800 FOV + 168 overlap + 112 align)

7.1.3.5.2 Note that if the instrument start and stop times between OLI and TIRS are not properly synchronized there could be <u>multiple</u> partial scenes at the beginning and/or ending of the interval. Based on the start time values above TIRS collects should start/end roughly 5 seconds before/after OLI [(40.01 – 29.67)/2 =5.17]

7.1.3.5.3 Partial Scenes

A Full WRS-2 scene product with approximately ten percent overlap is defined as 180km in along-track direction. Additionally, the Field-of-View (FOV) offset for each instrument and the boresight misalignment is included in the minimum number of frames to ensure coverage. For OLI, this is 7001 frames and for TIRS, this is 2801 frames. Partial WRS-2 scenes are defined as anything less than a Full WRS-2 scene. For partial scenes, the scene center is computed from the image frame closest to the nominal WRS-2 scene center. In other words; for partial scenes with more than half a scene in length, the computed scene center is the "actual" WRS-2 scene center. For partial scenes with less than half a scene in length, the computed scene center is the point within the imagery which is closest to the WRS-2 scene center. For short partials that are at the start of an interval this would be at the center point of the first line and for partials that are at the end of an interval this would be the center point of the last line.

In addition, Landsat 8 has two instruments which are commanded on/off separately so there may be times when one sensor is collecting data over a WRS-2 coverage area where the other instrument is not. One likely scenario is that the instrument start and stop times between OLI and TIRS are not properly synchronized. e.g.) TIRS start time is more than 5 seconds before/after OLI. This could lead to "incidental" partials. Incidental partials are considered partials because the data from one of the instruments is a partial or does not exist; while data from the other instrument is full or partial over the same WRS-2 path/row. The table below defines when "incidental" partials would occur:

OLI	TIRS	Scene
Full	Full	Full
Full	Partial	Partial (incidental)
Full	None	Partial (incidental)
Partial	Full	Partial (incidental)
Partial	Partial	Partial
Partial	None	Partial (incidental)
None	Full	Partial (incidental)
None	Partial	Partial (incidental)

Scenes are classified as partials (incidental) even though one of the instruments may have the full WRS-2 coverage, with overlap. Currently, products are not made from incidental partials as a combined scene collect is considered a partial. In the future, a combined scene collect with a full scene for one instrument and partial for the other may be separated such that a product can be made from the full scene.

Partials from an OLI Only or TIRS Only collect are not considered incidental partials.

7.1.3.5.4 Minimum Scene Overlap

WRS-2 scenes are defined to be at least 180km long which means approximately 10% overlap from scene to scene. Due to the pushbroom FOV nature of the Landsat 8 instruments, the start of a scene in Level-0 format needs to begin much earlier and end much later than past satellites to achieve 10% overlap in all bands on all Sensor Chip Assemblies (SCA). As illustrated in the following diagram, the minimum overlap requires 5% from each scene and ½ the FOV SCA staggering from each scene (See Appendix A Figure 9). In other words, the minimum number of frames for overlap is 10% of the center-to-center frame requirement + the total FOV SCA staggering requirement. In addition, to assure minimum overlap of scenes from different orbits, the alignment counts are factored into the minimum overlap for TIRS.

For OLI, the center-to-center requirement is 5664 frames, making the 10% minimum overlap requirement 566 frames and the FOV SCA staggering is 756 frames. So the minimum start-to-stop overlap is 566+756 = 1322 frames.

For TIRS, the center-to-center requirement is 1680 frames, making the 10% overlap requirement 168 and the FOV SCA staggering is 800 (including the allowance for science row deselect). Plus, the OLI-to-TIRS alignment adjustment is 112 frames. This makes the minimum start-to-stop overlap 168+800+112 = 1080 frames.

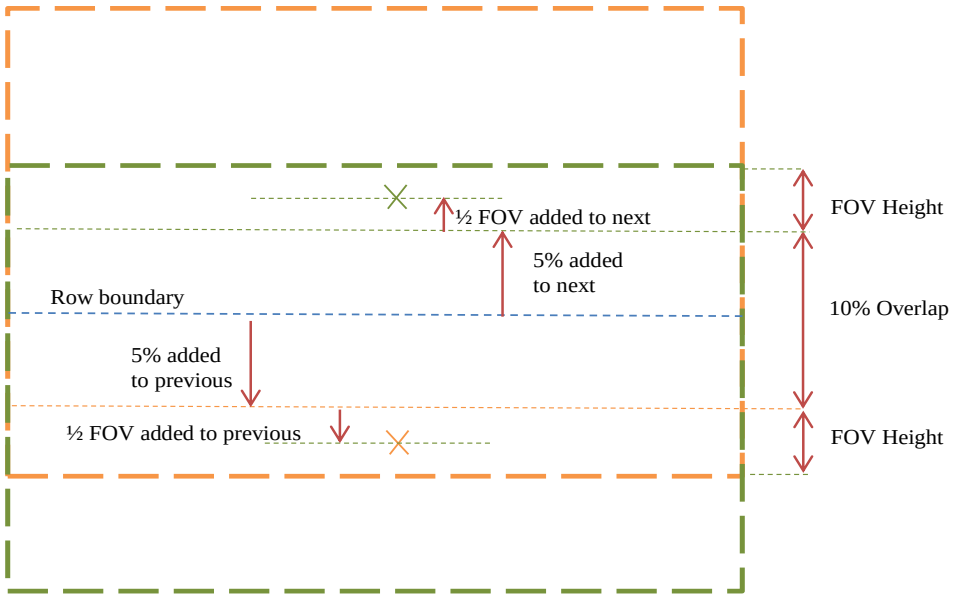


Figure 3 Minimum Overlap

7.1.3.5.5 Boresight Center

When the scene centers are determined from the LOS model, a boresight center is approximated from LOS projections of nearby detectors.

For OLI, the panchromatic band is the innermost band in the FOV and SCA 7 & 8 are the closest SCAs to the center. So, the left-most detector on SCA 8 and the rightmost detector on SCA 7 are used to average the latitude and longitude values obtained from the LOS projection. In the following diagram, the red dot marks the estimated boresight center using the projection of the two black detectors.

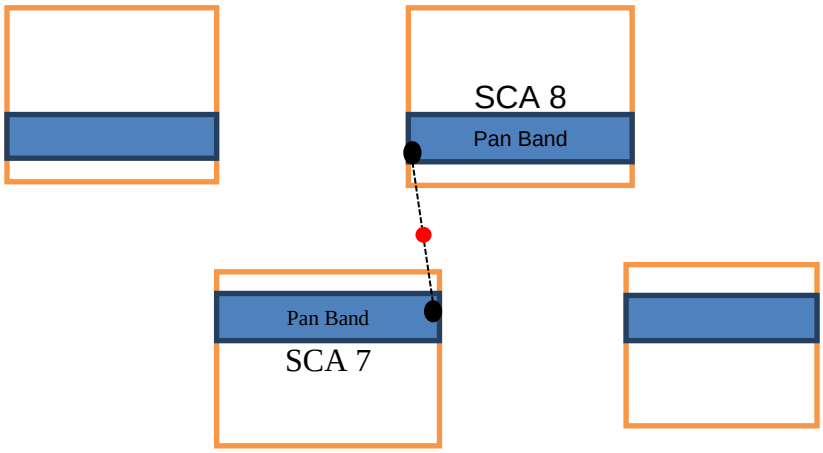


Figure 2 OLI Boresight Center

For TIRS, since the focal plane is made up of an odd number of SCAs, the boresight is first estimated by averaging two LOS latitude/longitude values from the outside two SCAs, then averaging the result with the center detector from the center SCA. In the following diagram, the blue dot represents the average of the outside two detectors and the red dot shows the average with the center SCA's detector, yielding the boresight center estimate.

Note that the boresight estimate will vary with the actual rows selected for each SCA and will vary with latitude. Near polar collects use an alternate method defined below.

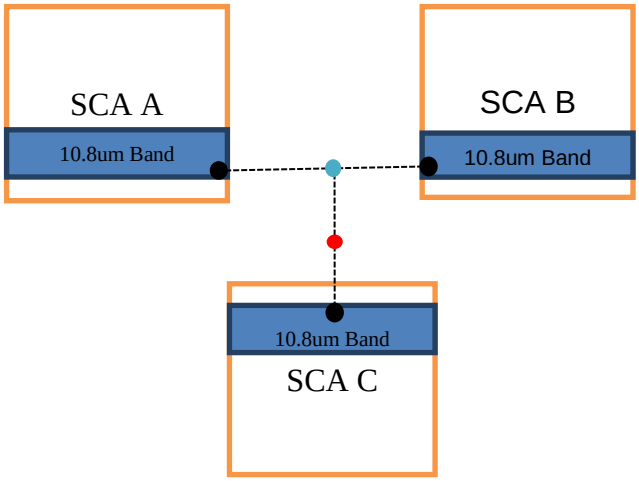


Figure 3 TIRS Boresight Center

7.1.3.6 Procedure

The primary tasks performed by the scene framing algorithm are to:

1. Load and preprocess the ancillary ephemeris and attitude data, and determine the image interval time span.
    a. The spacecraft ephemeris and attitude data from the interval ancillary data stream is quality checked and prepared for subsequent use by the ancillary data preprocessing algorithm. This algorithm is described in the Ancillary Data Preprocessing Algorithm Description Document (ADD).
    b. Load the interval image time codes for both instruments, if present, and determine the imaging interval start and stop times.
    c. Verify that the preprocessed ancillary data completely covers the imaging interval and this is at least 8 seconds of ancillary data before and after the image data. Note: 8 seconds is expected operationally but 4 seconds may be sufficient for processing.
2. Compute/Identify Orbital WRS-2 path/row (also known as nadir path/row) coverage within the imaging interval.
    a. Determine the ancillary time closest to the beginning of imaging, but not after. The ancillary time should be the latest of ACS, Ephemeris, and IMU times and the imaging time should be the earliest of OLI and TIRS image start times.
    b. Determine the ancillary time closest to the end of imaging, but not before. The ancillary time should be the earliest of ACS, Ephemeris, and IMU times and the imaging time should be the latest of OLI and TIRS image end times.
    c. Use the heritage nadir scene framing algorithm Determine Nadir WRS-2 Path/Row Sub-Algorithm below and the preprocessed ancillary data to compute the starting and ending fractional WRS-2 scene path/row values. The rounded values define the Orbital WRS-2 path/row span of the interval. These orbital path/rows are used for scene and interval

ids even for off-nadir imaging and for determining how the scene center times are computed.

d. Loop through the identified rows and use the heritage nadir scene framing algorithm Determine Nadir WRS-2 Path/Row Sub-Algorithm below and the preprocessed ancillary data to find the times where the fractional WRS-2 scene row values are whole numbers, i.e. where frow ≅ int(frow). These times define the initial nadir scene center times for each row.

e. Note: Because the above calculations are only done based on ancillary and ancillary data is captured before and after the imagery, the first and last scene center times calculated might fall outside the imagery collected. These center times are adjusted below.

3. Adjust scene center times within the rows found. The initial nadir scene center times are adjusted in the following ways:

   a. For non-polar region rows (Orbital WRS-2 rows from 5 through 115 and from 129 through 239)[3]; the scene center times are adjusted based on the OLI boresight[4] location.

      i. Define the OLI boresight line-of-sight vector as: $[\ 0\ 0\ 1\ ]^T$

      ii. Use the preprocessed ancillary data to interpolate the ECEF attitude quaternion at the scene center time as described below in the Interpolate Attitude Quaternion Sub-Algorithm. Note that the scene center ECEF ephemeris will already have been computed by the nadir scene framing algorithm.

      iii. Project the OLI boresight, to the WGS84 ellipsoid surface (i.e. using height = 0), using the algorithm described in the Forward Model, Get LOS Sub-Algorithm section of the "OLI Line-of-Sight Projection/Grid Generation Algorithm Description Document".

      Note:

      - Using the nadir scene center time, found above, allows us to bypass step a).1. Find Time,
      - Defining the OLI boresight LOS in step i. above takes the place of a).2. Find LOS,
      - The quaternion interpolation of step ii. above replaces a).3. Find Attitude,
      - Steps a).4. through a).7. of the Get LOS sub-algorithm can then be used to compute the geodetic latitude and longitude of the boresight intersection point.

      iv. Compute the (fractional) WRS-2 path/row corresponding to the boresight latitude/longitude using the Convert Geodetic Latitude/Longitude to WRS-2 Path/Row Sub-Algorithm described below.

[3] Note that these rows were selected as the boundary of the polar region because a 15 degree roll at this latitude corresponds to approximately a one row offset from nadir.
[4] Use the TIRS boresight if OLI isn't present.

v. Round the row to the nearest integer. This is the adjusted row to determine a new scene center time from.

vi. Use the Search for Scene Center Time Sub-Algorithm described below to adjust the scene center time until the boresight intersection point matches the scene center adjusted row.

vii. Declare the scene center time to be the new scene center.

b. For polar rows (Orbital WRS-2 rows122 and 246); the zero-crossing time of the z-component of the velocity vector from the ECEF ephemeris is the new scene center time. Note that this method will probably be close, if not the same, to the initial nadir scene center time, but is preferred implementation for accuracy and historical purposes.

c. For polar region rows (six rows adjacent to either side of the polar rows: Orbital WRS-2 rows 1-4, 116-121, 123-128, 240-245 and 247-248); the initial nadir scene center times will not be adjusted due to the large amount of overlap of path/rows at the pole. The new scene center time will be set to the nadir scene center time found above.[5]

4. Compute the scene extents

a. Determine the scene center frames for each instrument. If the first scene's center is before the start of imagery, a negative frame number is estimated. If the last scene's center is after the end of imagery, a frame number larger than the interval length is used. These are temporary values used in the next step.

b. Compute scene start and stop frames:

i. start $frame_i$ = max(0, center_$frame_i$ – (NOMINAL_SIZE/2)).

ii. stop $frame_i$ =

min(total_frames, center_$frame_i$ + (NOMINAL_SIZE/2)).

c. Adjust the first and last scene center frames to be within the actual imagery, if needed (i.e. if first center < 0, make it 0 and if the last center > interval last frame, make it the last frame).

d. Check for adequate overlap between scenes and adjust if needed:

```
overlap = scene stop frame_i - scene start frame_i+1.
if( overlap < minimum overlap )
    delta = (minimum overlap – overlap)
    start diff = (delta) / 2;
    if( start diff >= scene start frame_i+1 )
       start diff = scene start frame_i+1;
    scene start frame_i+1 -= start diff;
    stop diff = (delta – start diff);
    scene stop frame_i += stop diff;
    if ( scene stop frame_i > total_frames )
      scene stop frame_i = total_frames;
```

[5] Note that the number of adjacent rows may have to be adjusted if the polar region boundary scene sizes do not frame to the proper sizes.

e. Check to see if the first and last partials are completely within the overlap regions of adjacent scenes. If so, remove them.

f. Set the scene start, stop, and center times to the corresponding frame times.

g. Determine which scenes are partial or full.

   i. length = stop frame$_i$ – start frame$_i$;

   ii. if (length < NOMINAL_SIZE)
   scene is PARTIAL
   else
   scene is FULL

   iii. Note that for combined (OLI + TIRS) collects, both lengths must be greater than or equal to the corresponding nominal sizes for the scene to be FULL.

5. Check the complete list of scene center times to ensure that no two adjacent scene centers are more than 48 seconds apart (two times the normal scene center-to-center interval). If any two consecutive scene centers exceed this limit error out (the polar region will need to be enlarged). **This should never happen.**

6. Compute the corresponding *target* WRS-2 path/row coordinates and lat/long for each adjusted scene center time in the interval (new scene center times from step 3-4 above).

   a. Define the OLI boresight line-of-sight vector as: $[\ 0\ 0\ 1\ ]^T$

   b. Use the preprocessed ancillary data to interpolate the ECEF attitude quaternion at the adjusted scene center time as described in the Interpolate Attitude Quaternion Sub-Algorithm below.

   c. Project the OLI boresight, to the WGS84 ellipsoid surface (i.e. using height = 0), using the algorithm described in the Forward Model Get LOS Sub-Algorithm section of the "OLI Line-of-Sight Projection/Grid Generation Algorithm Description Document".

   Note:

      i. using the adjusted scene center time, found above, allows us to bypass step a).1. Find Time,

      ii. defining the OLI boresight LOS in step i. above takes the place of a).2. Find LOS,

      iii. the quaternion interpolation of step ii. above replaces a).3. Find Attitude,

      iv. Steps a).4. through a).7. of the Get LOS sub-algorithm can then be used to compute the geodetic latitude and longitude of the boresight intersection point.

   d. Compute the (fractional) WRS-2 path/row corresponding to the boresight latitude/longitude using the Convert Geodetic Latitude/Longitude to WRS-2 Path/Row Sub-Algorithm described below. Round the values to the nearest integers. This is the target path/row.

   e. Determine if the scene center position is off the WRS-2 grid. For collections near the poles, it is possible to look off-nadir toward the pole, into an area not defined by the WRS-2 grid. If the geodetic latitude just calculated is above 82.61 degrees this is considered as being off the WRS-2 grid and a special naming convention is used. To allow unique target row assignments, the North Pole area is assigned a row of 88n, and the South Pole area is assigned a row of 99n, where n is a sequential number. Up to

seven scenes can be covered in these areas; therefore, the scenes are assigned target row numbers 880 to 886, or 990 to 996 in the interval.

Corner Coordinate Framing

The corner points represent the WRS-2 extent of a scene on the ground in north-up latitude and longitude coordinates. Using the scenes starting and ending frames, found above, a Line-of-Sight is calculated at the first and last pixel in those frames (use the Forward Model Get LOS Sub-Algorithm section of the "OLI Line-of-Sight Projection/Grid Generation Algorithm Description Document" and "TIRS Line-of-Sight Projection/Grid Generation Algorithm Description Document). Due to the layout of the bands and SCAs on the focal plane, there are along-track offsets between bands within each SCA, along-track offsets between even and odd SCAs, and a reversal of the band ordering in adjacent SCAs. To create more uniform image coverage, the leading and trailing imagery associated with these offsets is "trimmed" based on an active area bounds.

To account for band offsets the frames and pixels from the outer most bands should be used in the corner calculations. For the OLI corner calculations, the Cirrus band is used and similarly the 10.8 µm band is used for TIRS. Using the outer most bands insures that every band is bounded by the corner coordinates. To account for the SCA offsets a minbox representing a rectangular active image frame is defined that excludes the excess trailing imagery from even SCAs and the excess leading imagery from odd SCAs.

OLI Active Image Area

The active image area (minbox) for OLI is computed by constructing 8 critical SCA corner points from the Cirrus band, labeled C1 through C8 in the figure below. Points C1 and C2 define the top edge of the active area, C3 and C4 the right edge, C5 and C6 the bottom edge, and C7 and C8 the left edge. Note that points C1 and C8 are the same (the upper left corner of SCA01) as are points C4 and C5 (the lower right corner of SCA14). Use the forward model to project these 8 line/sample locations to object space, computing the latitude/longitude coordinates for each point. The average elevation over the WRS-2 path/row is used as a rough adjustment from the WGS84 ellipsoid in the elevation parameter of the forward model for the 8 line/sample to latitude/longitude calculations. Use the WRS-2 Scene average elevation look-up file to determine the average elevation for the path/row being processed.

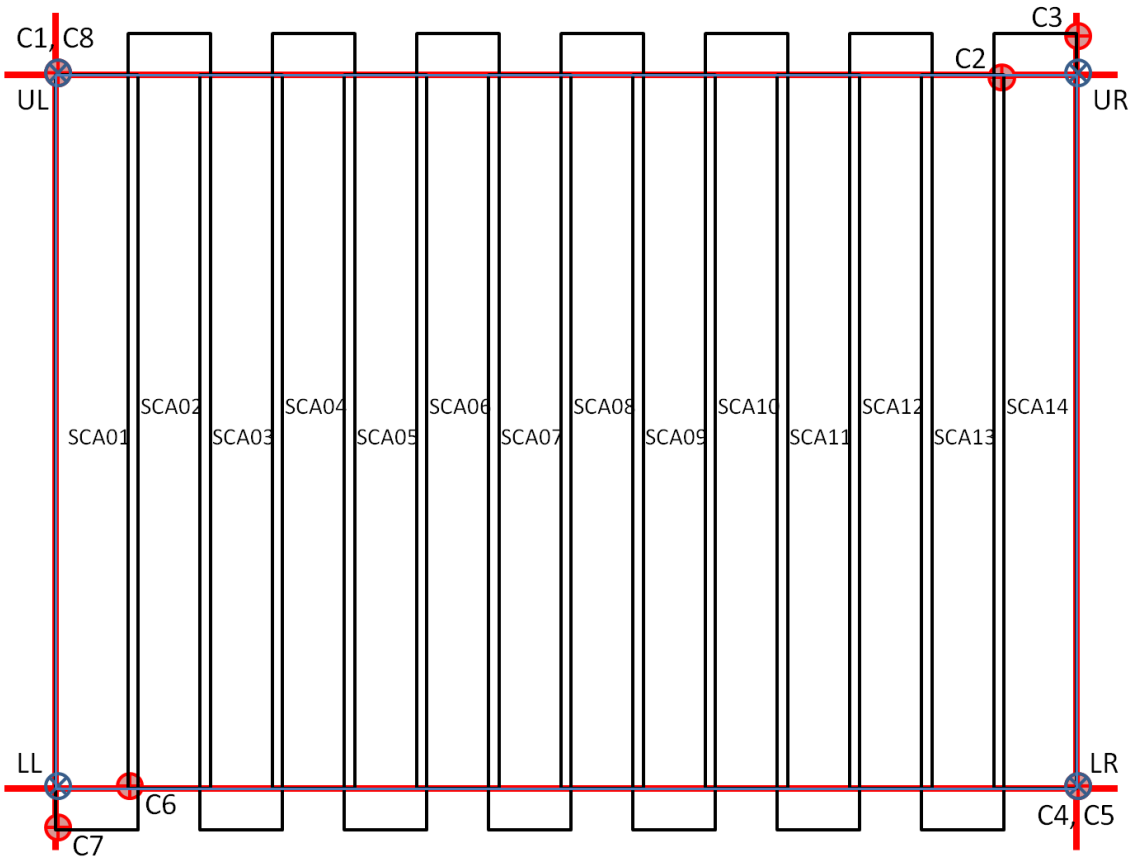


Figure 4 Active OLI Image Area

The remainder of the calculations and determination of the minbox framing of the active OLI image area is described in the Calculating the Active Image Area section of the "OLI Line-of-Sight Projection/Grid Generation" ADD. The results of these calculations should be the latitude and longitude of the four bounding corner points represented by the blue points in Figure 4.

TIRS Active Image Area

The active image area (minbox) for TIRS is computed by constructing 8 critical SCA corner points from the 10.8 µm band, labeled C1 through C8 in the figure below. This figure depicts the current understanding of the TIRS field of view orientation with respect to object space, but the algorithm described here will work so long as the SCAs are numbered sequentially across the field of view, in either direction. Points C1 and C2 define the top edge of the active area, C3 and C4 the right edge, C5 and C6 the bottom edge, and C7 and C8 the left edge. Note that points C4 and C5 are the same (the lower right corner of SCA01) as are points C6 and C7 (the lower left corner of SCA03). Use the forward model to project these 8 line/sample locations to object space, computing the latitude/longitude coordinates for each point. The average elevation over the WRS-2 path/row is used as a rough adjustment from the WGS84 ellipsoid in the elevation parameter of the forward model for the 8 line/sample to latitude/longitude calculations. Use the WRS-2 Scene average elevation look-up file to determine the average elevation for the path/row being processed.

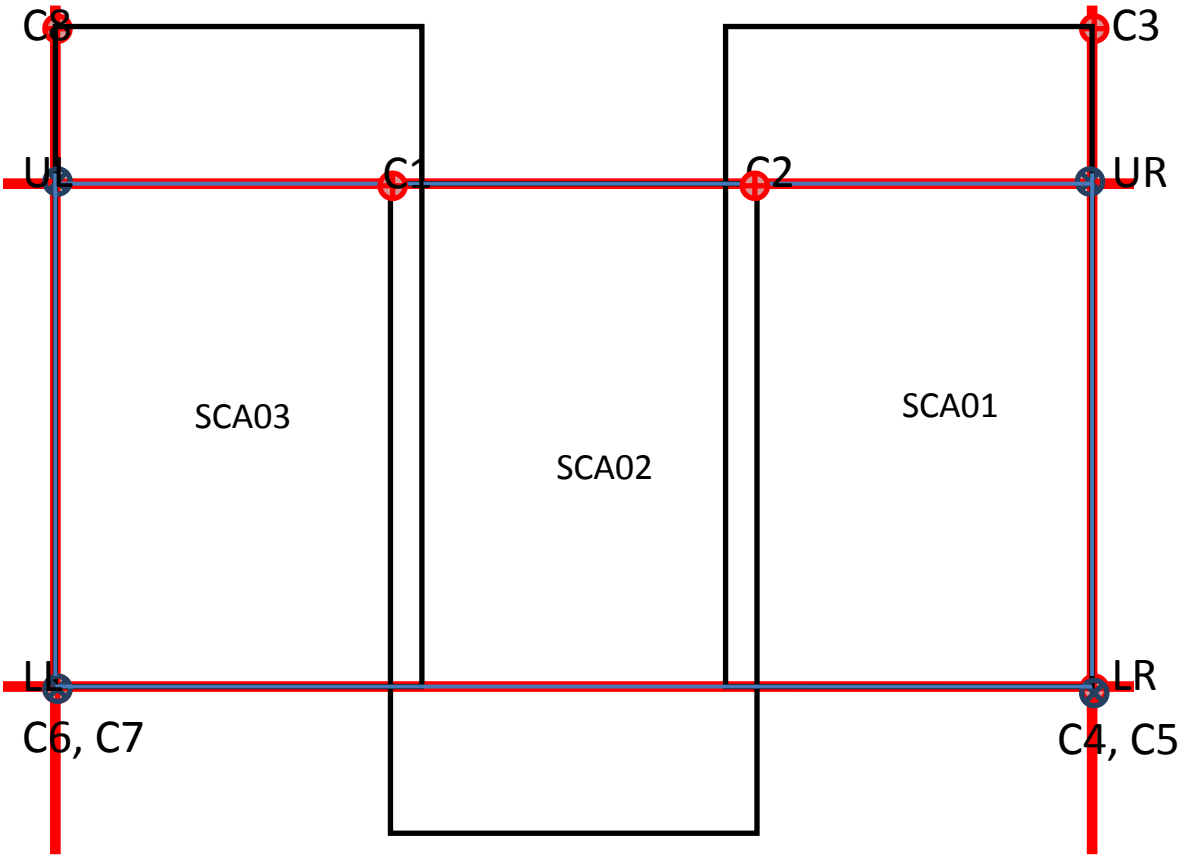


Figure 5 Active TIRS Image Area

The corner point assignments are made by examining the SCA across-track and along-track Legendre coefficients to determine: 1) whether SCA01 is on the left (+Y) or right (-Y) side of the scene; 2) whether even or odd SCAs lead; and 3) whether the sample number increases in the –Y or +Y direction. If the across-track Legendre constant term (coef_y0) for SCA01 is positive then it is the left-most SCA and SCA03 is the right-most. If the along-track Legendre constant term (coef_x0) for SCA01 is greater than that for SCA02, then the odd SCAs lead. If the across-track Legendre linear term (coef_y1) for SCA01 is negative, then the sample number increases in the –Y direction.

Having determined the orientation of the SCAs, assign the top edge to the left-most leading SCA upper left (UL) corner and the right-most leading SCA upper right (UR) corner, the right edge to the right-most SCA UR and lower right (LR) corners, the bottom edge to the right-most trailing SCA LR corner and left-most trailing SCA lower left (LL) corner, and the left edge to the left-most SCA LL and UL corners. As shown in the figure, for the TIRS: C1 = SCA02 (left-most leading SCA) UL, C2 = SCA02 (right-most leading SCA) UR, C3 = SCA01 (right-most SCA) UR, C4 = SCA01 (right-most SCA) LR, C5 = SCA01 (right-most trailing SCA) LR, C6 = SCA03 (left-most trailing SCA) LL, C7 = SCA03 (left-most SCA) LL, and C8 = SCA03 (left-most SCA) UL. Note that these assignments are based on the current TIRS SCA ordering of SCA-B = SCA01, SCA-C = SCA02, and SCA-A = SCA03, and could change if the SCA numbering system is revised. If this were to happen, the change would be reflected in the Legendre coefficients, so the logic described here would automatically compensate.

The remainder of the calculations and determination of the minbox framing of the active TIRS Image area is described in the Calculating the Active Image Area section of the "TIRS Line-of-Sight Projection/Grid Generation" ADD. The results of these calculations should be the latitude and longitude of the four bounding corner points represented by the blue points in Figure 5.

As depicted in Figure 6, the lower corner coordinates correspond to the leading edge (last line) of a scene, and upper coordinates correspond to the trailing edge (first line) of a scene from the outer most bands on the SCAs. Leading/Trailing edges are based on which SCA/Band/Detectors are first/last in relation to the direction of flight (ascending or descending) relative to the ground.

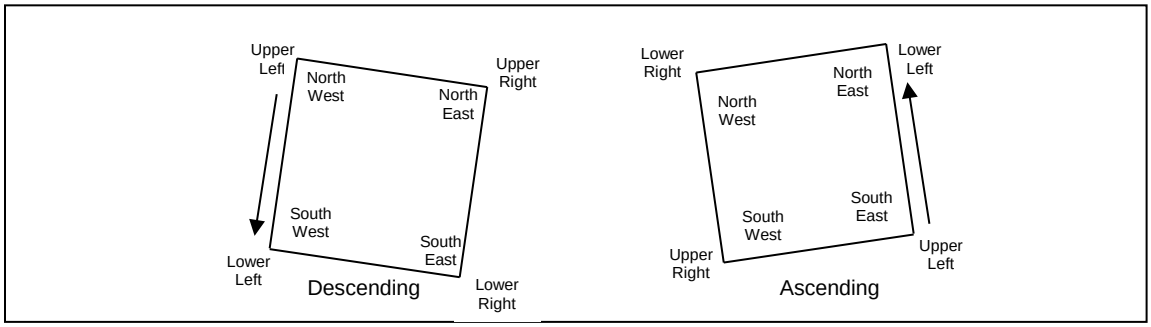


Figure 6 Leading/Trailing Scene Edge

Determine Nadir WRS-2 Path/Row Sub-Algorithm

The ephemeris data are used to define the nadir WRS-2 path & row. The following routine is called to determine the nadir pointing position of the satellite for Landsat Scene IDs and to determine scene center times for polar region rows. This is also known as the "heritage nadir scene framing algorithm".

Inputs:

- *ecef_pos, ecef_vel* (Ephemeris State Vector in Earth-Centered, Earth-Fixed coordinates).
- CPF WRS-2 Constants:
  - $Long\_Path_1\_Row_{60}$ (longitude of Path 1 at Row 60 = -64.6 deg).
  - *WRS_Cycle_Days* (number of days per WRS cycle = 16 days)
  - *WRS_Cycle_Orbits* (number of orbits per WRS cycle = 233 orbits)
  - *Scenes_Per_Orbit* (number of scenes or rows in each orbit = 248 rows)
  - *Descending_Node_Row* (row number of equator when descending = 60)
  - *Omega_E* (WGS-84 Earth's inertial rotation rate, rad/sec)

Outputs:

- Fractional WRS-2 Orbital Path & Row.

Procedure:

1. Convert the CPF path 1 row 60 longitude to radians.

$$Long\_Path_1\_Row_{60} = \frac{Long\_Path_1\_Row_{60} * \pi}{180}$$

2. Compute Earth angular rate (solar to account for orbital precession).

$$Earth\_Spin\_Rate = \frac{2*\pi}{24*3600}$$

3. Compute the spacecraft angular rate.

$$SC\_Ang\_Rate = \frac{2*\pi*WRS\_Cycle\_Orbits}{WRS\_Cycle\_Days*24*3600}$$

4. Normalize the incoming position and velocity vectors.

$$mag = \sqrt{ecef\_pos_x{}^2 + ecef\_pos_y{}^2 + ecef\_pos_z{}^2}$$

$$sc\_pos._x = \frac{ecef\_pos_x}{mag}$$

$$sc\_pos_y = \frac{ecef\_pos_y}{mag}$$

$$sc\_pos_z = \frac{ecef\_pos_z}{mag}$$

Adjust the velocity vector by earth's inertial rotation rate. NOTE: if the ephemeris data has already been preprocessed, the ADP output ECEF ephemeris can be used and this velocity adjustment isn't needed.

$$new\_vel._x = ecef\_vel_x - omega\_e*ecef\_pos_y$$

$$new\_vel._y = ecef\_vel_y + omega\_e*ecef\_pos_x$$

$$new\_vel._z = ecef\_vel_z$$

and normalize.

$$mag = \sqrt{new\_vel_x{}^2 + new\_vel_y{}^2 + new\_vel_z{}^2}$$

$$sc\_vel_x = \frac{new\_vel_x}{mag}$$

$$sc\_vel_y = \frac{new\_vel_y}{mag}$$

$$sc\_vel_z = \frac{new\_vel_z}{mag}$$

5. Compute the spacecraft angular momentum $R \times Y$.

$$ang\_mo = |sc\_pos \times sc\_vel|$$

6. Compute the vector to the descending node

$$dnode_x = \frac{ang\_mo_y}{\sqrt{ang\_mo_x^{\,2} + ang\_mo_y^{\,2}}}$$

$$dnode_y = \frac{-ang\_mo_x}{\sqrt{ang\_mo_x^{\,2} + ang\_mo_y^{\,2}}}$$

$$dnode_z = 0$$

and normalize.

$$mag = \sqrt{dnode_x^{\,2} + dnode_y^{\,2} + dnode_z^{\,2}}$$

$$dnode_x = \frac{dnode_x}{mag}$$

$$dnode_y = \frac{dnode_y}{mag}$$

$$dnode_z = \frac{dnode_z}{mag}$$

7. Compute the central travel angle from the descending node and the spacecraft position vector.

$$temp = dnode \times sc\_pos$$

$$mag = |temp|$$

$$s = \text{sign}(temp \bullet ang\_mo)$$

$$central\_angle = \text{atan}\,2(mag * s, (dnode \bullet sc\_pos))$$

8. Compute the row number from the central angle.

$$frow = Descending\_Node\_Row + \frac{central\_angle}{2*\pi} * Scenes\_Per\_Orbit$$

$$frow < 0.5 \Rightarrow frow = frow + Scenes\_Per\_Orbit$$

$$frow > (Scenes\_Per\_Orbit + 0.5) \Rightarrow frow = frow - Scenes\_Per\_Orbit$$

9. Compute the longitude of the instantaneous descending node.

$$inst\_dnode\_long = \text{atan}\,2(-ang\_mo_x, ang\_mo_y)$$

$$long\_origin = inst\_dnode\_long + \frac{central\_angle * Earth\_Spin\_Rate}{SC\_Ang\_Rate}$$

10. Compute the path number from the longitude of row 60.

$$fpath = \frac{Long\_Path1\_Row60 - long\_origin}{2*\pi} * WRS\_Cycle\_Days + 1$$

$$(central\_angle < \frac{(0.5 - Descending\_Node\_Row) * 2 * \pi}{Scenes\_Per\_Orbit}) \Rightarrow fpath = fpath - 16$$

$$(fpath < 0.5) \Rightarrow fpath = fpath + WRS\_Cycle\_Orbits$$

NOTE: the (0.5 – Descending_Node_Row) is the distance in WRS rows from the start of the path (row 0.5) to the descending node (row 60).

11. Make sure the row number is in range.

while (frow < 0.5)
fpath = fpath – 16;
frow = frow + Scenes_Per_Orbit;

while (frow > Scenes_Per_Orbit + 0.5)
fpath = fpath + 16;
frow = frow – Scenes_Per_Orbit;

12. Make sure the path number is in range.

while (fpath < 1)
fpath = fpath + WRS_Cycle_Orbits;

while (fpath > WRS_Cycle_Orbits)
fpath = fpath – WRS_Cycle_Orbits;
frow = frow – Scenes_Per_Orbit;

Interpolate Attitude Quaternion Sub-Algorithm
Given a sequence of time stamped quaternions, ($\mathbf{q_i}$, t_i), and a time, t_0, at which the interpolated quaternion is desired:

1. Step through the quaternion time stamps to identify the latest quaternion time, t_i, which is less than or equal to the interpolation time of interest, t_0.
2. Calculate the quaternion $\mathbf{\Delta q}$ that rotates $\mathbf{q_i}$ to $\mathbf{q_{i+1}}$:
   $\mathbf{\Delta q} = \mathbf{q_{i+1}}\ \mathbf{q'_i}$
   where: $\mathbf{q'_i}$ is the conjugate of quaternion $\mathbf{q_i}$. See the quaternion conjugation and quaternion multiplication sub-algorithms below.
3. If the sign of the fourth element of $\mathbf{\Delta q}$, Δq_4, is negative, change the sign of the entire quaternion.
4. Decompose the $\mathbf{\Delta q}$ quaternion into its angle (θ) and axis of rotation ($\mathbf{x}$) form:
   $\sin(\theta/2) = \text{sqrt}(\Delta q_1 * \Delta q_1 + \Delta q_2 * \Delta q_2 + \Delta q_3 * \Delta q_3)$
   $\cos(\theta/2) = \Delta q_4$
   $\theta = 2 * \text{atan}(\sin(\theta/2) / \cos(\theta/2))$

$\mathbf{x} = [\ \Delta q_1/\sin(\theta/2)\ \ \Delta q_2/\sin(\theta/2)\ \ \Delta q_3/\sin(\theta/2)\ ]^T$
noting that if $\sin(\theta/2) = 0$ then $\mathbf{x} = \mathbf{0}$.

5. Linearly interpolate the angle θ_0 at time t_0:
   $\theta_0 = \theta * (t_0 - t_i) / (t_{i+1} - t_i)$
6. Construct the quaternion corresponding to the new rotation angle θ_0:
   $\mathbf{\Delta q_0} = [\ \sin(\theta_0/2)\ \mathbf{x}^T\ \ \cos(\theta_0/2)\ ]$
7. Apply the new delta quaternion to $\mathbf{q_i}$ to compute $\mathbf{q_0}$, the quaternion at time t_0:
   $\mathbf{q_0} = \mathbf{\Delta q_0}\ \mathbf{q_i}$

Quaternion Conjugation Sub-Algorithm
The conjugate **q'**, of a quaternion **q**, is computed by inverting the sign on the first three elements of **q**:

$q' = [\ -q_1\ \ -q_2\ \ -q_3\ \ q_4\ ]$

Quaternion Multiplication Sub-Algorithm
The product **c**, of quaternions **a** and **b** is given by:

$c_1 = a_4 b_1 + a_3 b_2 - a_2 b_3 + a_1 b_4$
$c_2 = -a_3 b_1 + a_4 b_2 + a_1 b_3 + a_2 b_4$
$c_3 = a_2 b_1 - a_1 b_2 + a_4 b_3 + a_3 b_4$
$c_4 = -a_1 b_1 - a_2 b_2 - a_3 b_3 + a_4 b_4$

Note that quaternion multiplication does not commute. Also note that other formulations of quaternion multiplication are possible. Any consistent formulation should work in the interpolation equations above.

Convert Geodetic Latitude/Longitude to WRS-2 Path/Row Sub-Algorithm
Given a boresight geodetic latitude ϕ, and longitude λ, and the corresponding spacecraft velocity Z component V_z (to determine whether the scene is ascending or descending):

1. Compute the geocentric latitude θ, from the geodetic latitude ϕ, and the WRS84 ellipsoid semi-major (a) and semi-minor (b) axes:

   $\theta = \text{atan}(\ \tan(\ \phi\ ) * b/a * b/a\ )$

2. Use the geocentric latitude and the nominal Landsat orbital inclination (i = 98.2 degrees) to compute the longitude offset to the apparent descending node:

   $\lambda_{off} = \text{asin}(\ \tan(\ \theta\ ) / \tan(\pi - i)\ )$

   This calculation should include a test to ensure that the argument of the asin function is in the range -1 to +1, clipping the value to this range if necessary (e.g., for latitudes outside the standard WRS-2 range).

3. Calculate the central travel angles for both the descending and ascending cases:

   $CTA_d = \text{asin}(\ -\sin(\ \theta\ ) / \sin(\pi - i)\ )$
   $CTA_a = \pi - CTA_d$

   As above, the range of the asin function argument should be clipped to the range -1 to +1.

4. Compute the allowable range of central travel angles on a given WRS path as:

   min CTA = $(0.5 - 60.0)/248 * 2\pi$
   max CTA = $(248.5 - 60.0)/248 * 2\pi$

5. Add or subtract 2π to the descending and ascending central travel angles to bring them within the allowable range.

   If (CTA_d < min CTA) $CTA'_d = CTA_d + 2\pi$
   Else if (CTA_d > max CTA) $CTA'_d = CTA_d - 2\pi$
   Else $CTA'_d = CTA_d$
   If (CTA_a < min CTA) $CTA'_a = CTA_a + 2\pi$
   Else if (CTA_a > max CTA) $CTA'_a = CTA_a - 2\pi$
   Else $CTA'_a = CTA_a$

6. Compute the Earth rotation angles from the adjusted central travel angles:

   $\Delta\lambda_d$ = CTA'_d * Earth rotation rate / Spacecraft angular rate
   $\Delta\lambda_a$ = CTA'_a * Earth rotation rate / Spacecraft angular rate

7. Calculate the apparent descending node longitudes for the descending and ascending cases:

   $DN\lambda_d = \lambda - \lambda_{off} + \Delta\lambda_d$
   $DN\lambda_a = \lambda + \lambda_{off} + \pi + \Delta\lambda_a$

8. Select the descending or ascending case:

   If (Vz > 0.0)
   CTA' = CTA'_a
   $DN\lambda = DN\lambda_a$
   Else
   CTA' = CTA'_d
   $DN\lambda = DN\lambda_d$

9. Compute the (fractional) WRS-2 row number from the central travel angle:

   row = row at equator (60) + CTA' / 2π * number of rows (248)

10. Compute the (fractional) WRS-2 path number:

    λ_0 = (longitude of path 1 at equator) - $DN\lambda + 2\pi$
    If ($\lambda_0 > 2\pi$) $\lambda_0 = \lambda_0 - 2\pi$
    path = λ_0 * number of paths (233) / 2π + 1

Search for Scene Center Time Sub-Algorithm
Given a scene center time (t_0) and the corresponding WRS-2 row (row_0) and a target (integer) row (row_T):

1. Compute the nominal WRS-2 row rate:

   row rate = number of rows (248) / orbital period

2. Compute the row error:

   row error = $row_0 - row_T$

3. If the absolute value of the row error is less than 0.005, use the current scene center and exit the search.

4. Save the previous scene center time and row:

   $row_L = row_0$

$t_L = t_0$

5. Adjust the scene center time:

   $t_0 = t_0$ - (row error) / (row rate)

6. Interpolate the spacecraft ephemeris and attitude at the new scene center time.
7. Project the boresight to the WGS84 ellipsoid at the new scene center time.
8. Compute the WRS-2 path/row at the boresight latitude/longitude as described above. This yields a new value of row_0.
9. Compute the WRS-2 row rate:

   row rate = $(row_0 - row_L) / (t_0 - t_L)$

10. Continue the iterations at step 2. above.

Note: This sub-algorithm is only used for non-polar scenes so the row transition from 248 to 1 is not an issue.

7.1.3.7 Maturity

The Scene Framing Algorithm is an attempt to document how OLI and TIRS scene sizes were derived and how to determine scene centers, orbital WRS-2 path/rows and target WRS-2 path/rows. In addition it addresses where to calculate corner point information to form the coordinates of the WRS-2 frame for the metadata. This algorithm relies on invoking all or part of the ancillary data preprocessing and LOS projection algorithms.

Being that this document was written before launch, most of the OLI and TIRS instrument information is known for the supporting calculations. However, there may be changes due to results from various instrument and spacecraft tests. As more data is received and analyzed, appropriate changes will be made as necessary.

7.1.3.8 Notes

Significant algorithm assumptions and notes, including those embedded in the text above, are:

1. Ancillary data for the full imaging interval with 8 seconds of extra before and after the interval is available to provide the required geometric support data. A minimum of 4 seconds of extra ancillary data before and after **may** be sufficient for processing.
2. In the polar transition regions (rows 5, 115, 129, and 239) there may be larger scenes framed in situations where the spacecraft is rolled "away" from the pole (see step 2.d. of the main algorithm procedure above). In this configuration the distance between scene centers can exceed 5664 OLI multispectral image frames. This occurs because the OLI is looking toward the equator where the rows are growing farther apart in latitude, while the spacecraft is flying at higher latitude where the rows are closer together in latitude. It thus takes more than a nominal row of flight time for the boresight to traverse one row of latitude. Other possible approaches would be to move the transition region toward the equator by 5 rows for intervals that are rolled toward the equator. The approach adopted here is to allow the off-nadir L0Rp scenes to be somewhat longer than nadir scenes.

7.1.3.8.1 Appendix A – Scene Sizing Background

OLI:

The bore sight of the OLI telescope is parallel to the spacecraft +Z axis which means it will be nadir pointing. The 14 OLI SCAs are arranged in two rows of seven as shown in Figure 7. The Field of View (FOV) of the telescope is 15 degrees in the cross track direction and 1.7 degrees in the along-track direction.

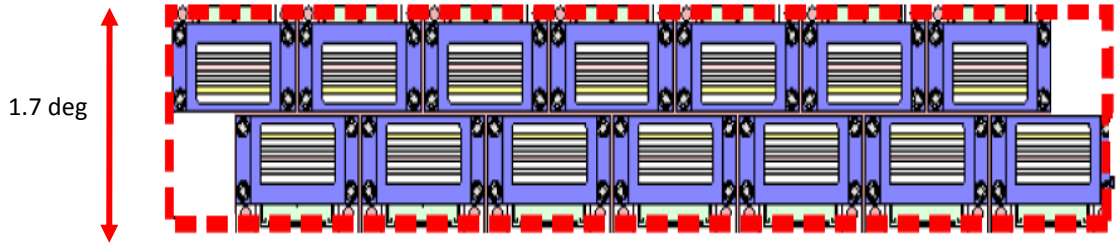


Figure 7 OLI SCA Layouts and FOV

The telescope bore sight will traverse one scene (i.e. Scene center to scene center) in about 24 seconds given the nominal orbit rate.

WRS_Cycle_Days = 16 days
WRS_Cycle_Orbits = 233
Scenes_Per_Orbit = 248
Seconds_Day = 86400

Calculate Spacecraft Angular Rotational rate:

$$\omega = \frac{2 * \pi * WRS\_Cycle\_Orbits}{WRS\_Cycle\_Days * Seconds\_Day} = 0.0010591049 \frac{radians}{seconds}$$

Calculate time between successive WRS rows:

$$\Delta\, t = \frac{WRS\_Cycle\_Days\ * Second\_Day}{WRS\_Cycle\_Orbits * Scenes\_Per\_Orbit} = 23.923577\ seconds$$

The size of an OLI scene, in lines, can be calculated with respect to the bore sight with the following calculations. Rounding the above number to 24 seconds the number of OLI lines between successive WRS rows is:

OLI_Frame_Rate = 236 lines / second

$$OLI\ Lines = \frac{\Delta\, t}{OLI\_Frame\_Rate} = 5664\ Lines$$

In addition, by definition of Landsat WRS products, WRS scenes are overlapped with the previous row by 5% and the next row by 5%. The total number of OLI lines needed for the overlap is then 10%.

$$OLI\ Lines = OLI\ Lines + 0.1 * \frac{\Delta t}{OLI\_Frame\_Rate} = 6230\ Lines$$

However the LDCM OLI bands within each SCA are staggered with respect to the bore sight.

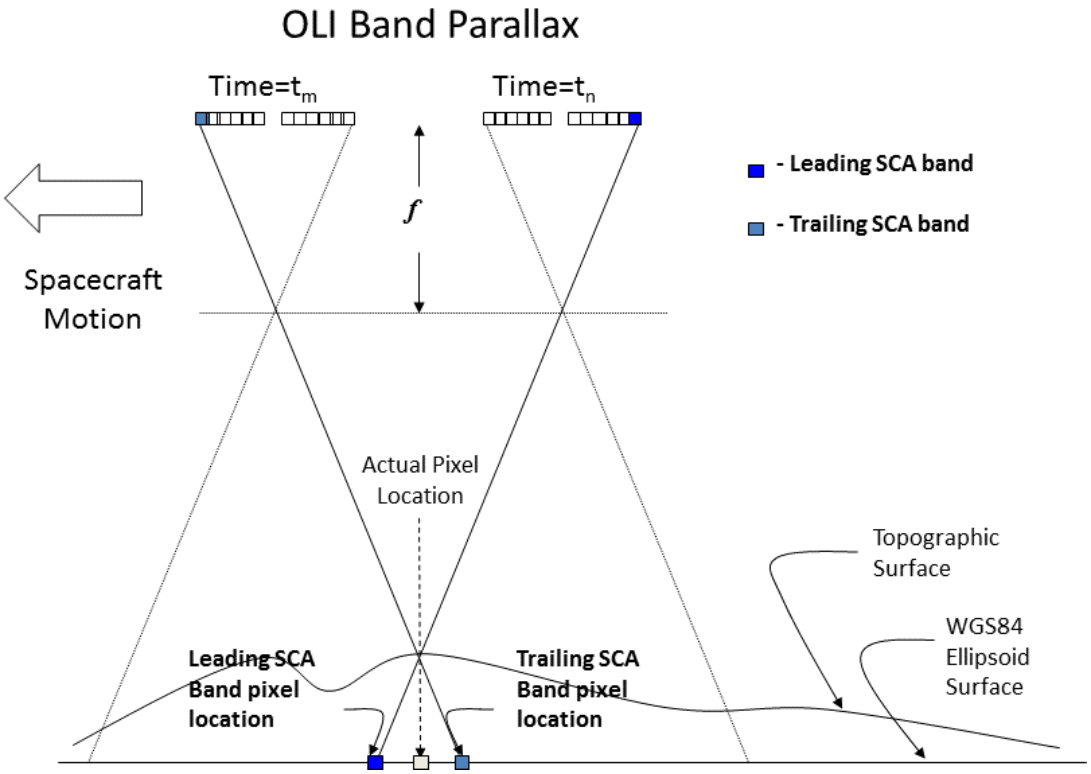


Figure 8 OLI Band/SCA Band Parallax

This staggering represents a time difference between when the leading set of detectors within a SCA, for a given band, image the start and end of the WRS scene and when the trailing set of detectors within a SCA, for a given band, image the start and end of the WRS scene.

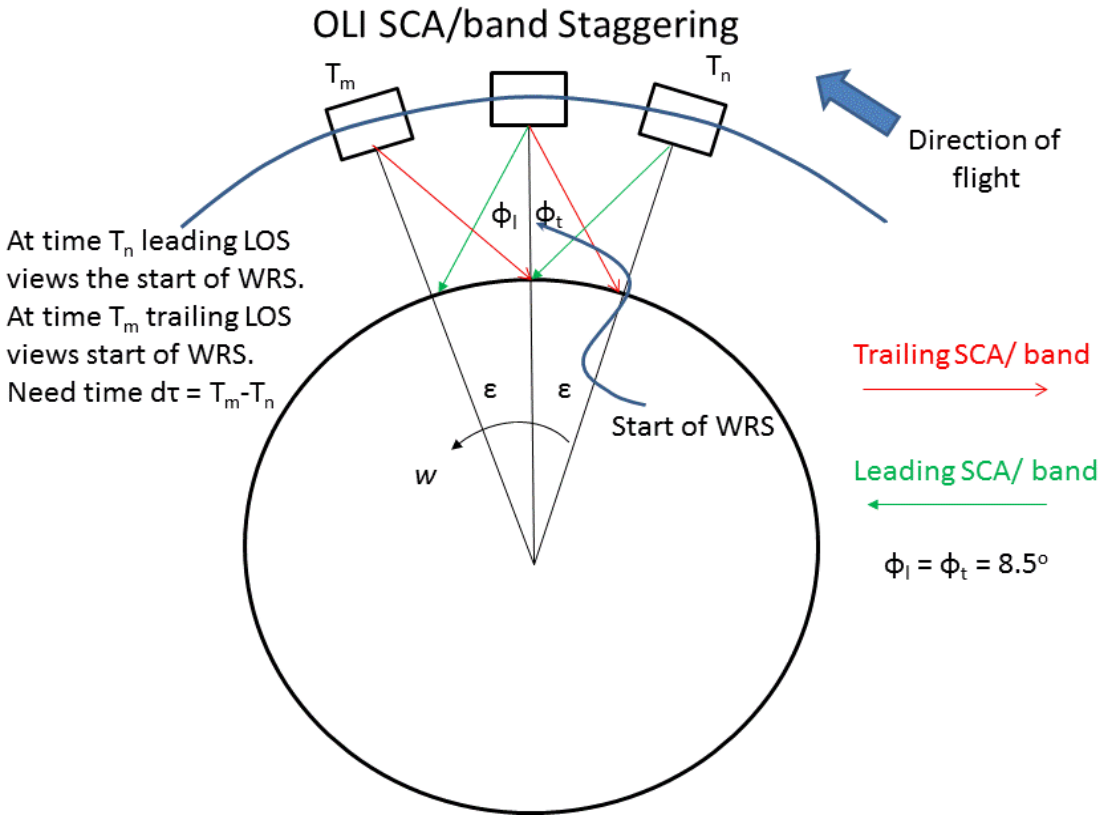


Figure 9 OLI SCA/Band Staggering

The extra time required for the leading and trailing imaging bands of the SCAs to cover a point on the ground relative to the center of the focal plane will vary with position in orbit and scene elevation. Extra lines of imaging will be required at the beginning and end of the interval, or about 1.5 seconds of extra data on each side. This means that imaging must start at least 1.5 seconds prior to the telescope bore sight reaching the leading edge of the first scene in the desired interval and must continue for at least 1.5 seconds after passing over the trailing edge of the last scene of an interval. This will assure that all bands in all SCAs have completely imaged the scene.

Looking at the OLI Legendre LOS polynomials and determining the leading and trailing look vectors, the difference in radians/degrees is:

Leading_LOS = 1.436414e-02 radians = 0.8230046 degrees
Trailing_LOS = -1.444101e-02 radians = -0.8265414 degrees

The difference between these two numbers represents a field of view of 1.65 degrees. Rounding this to 1.7 degrees (0.85 leading and 0.85 trailing) we can determine the amount of time needed to pad either the leading or trailing acquisition of the OLI WRS scene in terms of time. If the maximum satellite altitude is present at the poles and the minimum satellite altitude is present at the equator the minimum and maximum number of lines needed in the L0rp can be calculated.

α = 0.85 degrees

Orbital_Radius = 7083445,719 meters
Semi_Major_Axis = 6378137.0 meters
Semi_Minor_Axis = 6356752.3142 meters
Major_Alt = Orbital_Radius – Semi_Major_Axis = 705308.72 meters
Minor_Alt = Orbital_Radius – Semi_Major_Axis = 726693.40 meters

The ground distance covered at these two altitudes represented on the earth are found as:

$$GD\_Major = Major\_Alt * \tan\alpha = 10464.234 \ meters$$
$$GD\_Minor = Minor\_Alt * \tan\alpha = 10781.505 \ meters$$

These ground distances represent an angular orbit change of:

$$\varepsilon = \text{atan}\left(\frac{GD\_Major}{Semi\_Major\_Axis}\right) = 0.00164063 \ radians$$

$$\varepsilon = \text{atan}\left(\frac{GD\_Minor}{Semi\_Minor\_Axis}\right) = 0.00169607 \ radians$$

Using the spacecraft angular rotational rate gives a delta time due to the staggering of the SCAs as:

$$d\tau = \frac{\varepsilon}{\omega} = 1.5 \ and \ 1.6 \ seconds$$

Using the OLI frame rate the number of OLI lines needed for this (maximum) delta time is:

$$\Delta \ Lines \ SCA \ Staggering = 1.6 \ seconds * OLI\_Frame\_Rate = 378 \ Lines$$

The total number of lines needed within a OLI WRS scene is then

Total Lines = (nominal size + 5% overlap + SCA staggering)

$$OLI \ Lines = OLI \ Lines + 2 * \Delta \ Lines \ SCA \ Staggering = 6986 \ Lines$$

For convenience and ease of use the final number of OLI lines will be rounded to 7001 lines. An odd number is used so that a center line is found and the same number of before and after lines (3500) are used to define the entire L0R scene.

In terms of time, a single scene will take about 29.6 seconds:

1.6 sec (378 lines for leading edge SCA coverage)
\+ 1.2 sec (5% overlap on leading edge)
\+ 24.0 sec (time for WRS scene)
\+ 1.2 sec (5% overlap on trailing edge)
\+ 1.6 sec (378 lines for trailing edge SCA coverage)
= 29.6 sec (29.67 for 7001 lines)

TIRS:
The TIRS SCAs are larger and farther apart than the OLI SCAs which will require TIRS to begin imaging earlier, and continue for a longer duration than OLI in order to completely image a scene. The along-track of the TIRS instrument is 4.95 degrees (Figure 10) requiring an additional 9.43 seconds (4.714 seconds for the leading and trailing edges) of imaging to assure all bands in all three

SCAs have completely imaged the scene. The same logic used for calculating the number of OLI lines can be used for TIRS.

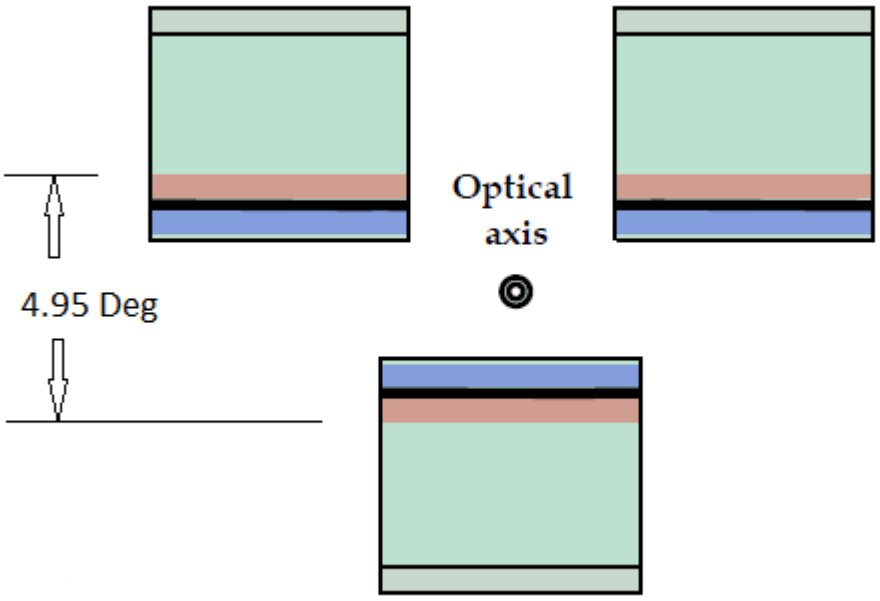


Figure 10 TIRS Layout and FOV

The size of a TIRS scene, in lines, can be calculated with respect to the bore sight with the following calculations. The number of TIRS lines between successive WRS rows:

TIRS_Frame_Rate = 70 lines / second

$$TIRS\ Lines = \frac{\Delta\ t}{TIRS\_Frame\_Rate} = 1680\ Lines$$

In addition, by definition of Landsat WRS products, WRS scenes are overlapped with the previous row by 5% and the next row by 5%. The total number of TIRS lines needed for the overlap is then 10%.

$$TIRS\ Line = TIRS\ Line + 0.1 * \frac{\Delta\ t}{TIRS\_Frame\_Rate} = 1848\ Lines$$

Looking at the TIRS Legendre LOS polynomials and determining the leading and trailing look vectors, the difference in radians/degrees is:

Leading_LOS = 4.623841e-02 radians = 2.65 degrees
Trailing_LOS = -3.989761e-02 radians = -2.29 degrees

The difference between these two numbers represents a field of view of 4.94 degrees. Rounding this to 5 degrees (2.7 leading and 2.3 trailing) we can determine the amount of time needed to pad either the leading or trailing acquisition of the TIRS WRS scene in terms of time. If the maximum satellite

altitude is present at the poles and the minimum satellite altitude is present at the equator the minimum and maximum number of lines needed in the L0rp can be calculated. Using the spacecraft angular rotational rate gives a delta time due to the staggering of the SCAs as:

$$d\tau = 4.92 \text{ seconds (maximum)}$$

Using the TIRS frame rate the number of TIRS lines needed for this (maximum) delta time is:

$$\Delta\ Lines\ SCA\ Staggering = 5\ seconds * TIRS\_Frame\_Rate = 350\ Lines$$

This delta is further extended by 50 lines to include the possible use of the secondary, or *science,* rows. The total number of lines needed within a TIRS WRS scene is then

Total Lines = (nominal size + 5% overlap + SCA staggering + 50 science)

$$TIRS\ Lines = TIRS\ Lines + 2 * (\Delta\ Lines\ SCA\ Staggering + 50) = 2648\ Lines$$

In terms of time, a single scene will take about 37.8 seconds:

5.0 sec (350 lines for leading edge SCA coverage)
\+ 0.7 sec (50 lines to include science rows)
\+ 1.2 sec (5% overlap on leading edge)
+ 24.0 sec (time for WRS scene)
\+ 1.2 sec (5% overlap on trailing edge)
\+ 0.7 sec (50 lines to include science rows)
\+ 5.0 sec (350 lines for trailing edge SCA coverage)
= 37.8 sec

To ensure TIRS fully covers OLI additional lines should be added to account for any misalignment between the OLI and TIRS bore sights (which has a one second tolerance) and OLI to ACS alignment, any biases present in the pointing of the Scene Select Mirror, etc. Also, the above calculations use nominal values, pre-launch information, and rounding liberties which should be taken into consideration.

The TIRS number of lines can also be calculated based on the number of OLI lines to ensure full coverage of the TIRS data with OLI data. The number of TIRS lines can be found by first scaling the OLI number of lines needed to cover the 180km scene (6230 lines) by the ratio of the nominal line sample rates of OLI-to-TIRS:

$$TIRS\ Lines = 6230\ OLI\ Lines * \frac{70\ Hz}{236\ Hz} = 1848\ Lines$$

This number then needs to be adjusted for the TIRS leading and trailing SCA/band staggering. From the above TIRS calculations this value is 700 lines. Another 112 lines are needed for the alignment tolerances and 100 lines to include the science rows. The total number of lines needed within a TIRS WRS scene that will fully cover the OLI data is then:

$$TIRS\ lines = 1848 + 700 + 100 + 112 = 2760\ Lines$$

For convenience and ease of use the final number of TIRS lines will be rounded to 2801 lines. An odd number is used so that a center frame is found and the same number of before and after frames (1400) are used to define the entire L0R scene.

This is equivalent to 40.01 seconds of TIRS data to cover the 29.67 seconds of OLI data. This means TIRS imaging will need to start/end approximately 5.2 seconds before/after OLI to ensure adequate coverage.

From the calculations listed above the size of the OLI will be 7001 lines (multispectral) and the size of the TIRS will be 2801 lines (thermal).

7.1.3.8.2 Appendix B – Comparison of Partial Scene Definitions

The Level-0ra DFCB for Landsat 7 has the following discussion of partial scenes:

> For a partial scene with more than half a scene length data, the computed "actual" scene center is also expected to happen in the proximity of the nominal WRS scene center. The "actual" scene center for a greater than half a scene length partial scene may also be computed from the available actual PCD and indexed to actual data in the band file. For a partial scene with less than half a scene length data, the scene center may have to be computed from extrapolated* ephemeris (no actual PCD may be available from the subinterval). The computed "imaginary" scene center for such a partial scene (less than half a scene) is still determined in the proximity of the nominal WRS scene center, but there will not be any actual band data in the subinterval band file for which to relate the scene center. The computed "imaginary" scene center for a partial scene with less than half a scene length data is indexed to an "imaginary scan" (non-existent scan 0) in the band file.
>
> * For LDCM, extrapolation of ancillary data beyond the existing ancillary/ephemeris was not provided and/or is not possible. The above method will not work for LDCM so it was decided that the scene center would actually represent the center of the partial (although the implementation currently takes the first or last line in the scene closest to the WRS-2 center. A future release will address this difference).

The current LMDD definition for LDCM partial scenes is:

> FULL = Full WRS scene - the standard 180 x 185km WRS size.
>
> PARTIAL = Partial WRS scene - less than full and includes the scene center, or greater than half and includes the scene center.

The definition provided in CCR#598 reads:

> PROPOSED: "...less than a full scene that is not covered by overlap."
>
> RATIONALE FOR CHANGE
>
> -provide consistency with heritage Landsat
>
> -have all scene metadata available in the Inventory
>
> -enable future ability to handle adhoc requests for partial scenes in Subsetter

The current DFCB definition for partial scenes is:

> Full – A full WRS scene product with approximately ten percent overlap is defined as 180km. For OLI, this is 7001 frames (~28.86-meter MS lines) and for TIRS, this is 2801 frames (~86.91-meter lines).
>
> Partial – Considered less than a full scene.
>
> Scene Center - The computed "actual" scene centers are from the image frame closet to the nominal WRS scene center

7.1.4 Ancillary Data Preprocessing Algorithm

7.1.4.1 Background/Introduction

The ancillary data preprocessing algorithm prepares the ancillary data provided by the spacecraft in the wideband data stream for use by subsequent geometric algorithms. This includes quality checking the incoming data to identify and remove outliers, applying scale factors from the CPF to convert the relevant ancillary data fields to engineering units, and processing the spacecraft attitude and ephemeris data to construct consistent attitude and ephemeris time histories for the data set. The baseline assumption is that the attitude and position/velocity estimates produced by the spacecraft will be sufficiently accurate to achieve LDCM geolocation accuracy requirements. If this is the case, only basic quality checking and smoothing operations will be required. Since the ancillary data stream will also include the raw observable data used by the spacecraft to construct its attitude and ephemeris estimates, more sophisticated processing using the raw star tracker and SIRU data to construct a “definitive” attitude data set, and/or using the raw GPS pseudo-range and carrier phase observables to compute a “definitive” ephemeris data set, would be possible. These enhanced capabilities are currently considered a risk reduction contingency that would not be implemented unless needed.

The content and structure of the spacecraft ancillary data is defined in the Space to Ground Station Interface Control Document. This document clarifies several uncertainties regarding formats, coordinate systems, and sampling rates that required assumptions to be made in earlier versions of this algorithm description. For example, the rate at which the integrated spacecraft attitude estimates are provided was initially undecided. Had the integrated spacecraft attitude estimate quaternions been provided at a lower rate than the SIRU data, those estimates would have required densification using the raw SIRU data and its associated calibration parameters, status flags, and on-board bias, alignment, and scale estimates. Since the degree of smoothing that the SIRU measurements will be subjected to in the on-board attitude filter is, as yet, unknown, this algorithm still assumes that the raw SIRU measurements will be used to ensure that high frequency content is preserved. The Smooth Euler and SIRU sub-algorithm will be used to perform this data blending and to replace quaternions flagged as outliers.

This algorithm was originally intended to support only imaging intervals that would be suitable for Level 1 processing – primarily Earth-view and lunar acquisitions. Subsequently, it was decided that ancillary data preprocessing would be valuable in other cases, particularly solar calibration intervals. Since these intervals tend to be quite short (only a few seconds) some special logic was added to allow processing to proceed under these conditions. These adjustments, mainly to the SIRU processing logic, are noted in the appropriate locations below.

7.1.4.2 Dependencies

The ancillary data preprocessing algorithm assumes that ancillary data for the full imaging interval with (nominally) 4 seconds of extra data before and after the interval, is available to provide the required geometric support data, that a CPF containing the scale factors needed to convert the ancillary data to engineering units is available, and that the quality thresholds needed to detect and remove or repair outliers are provided either in the CPF or as processing parameters.

7.1.4.3 Inputs

The ancillary data preprocessing algorithm and its component sub-algorithms use the inputs listed in the following table. Note that some of these "inputs" are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data; including data set IDs to provide unique identifiers for data trending).

Algorithm Inputs
ODL File (implementation)
CPF File Name
Relevant CPF contents:
Ancillary data engineering unit conversion factors
SIRU to ACS alignment matrix
SIRU engineering unit conversion factors
Leap Second Table
Ancillary data thresholds and limits
Orbital Radius Limits (nominal and max excursion)
Ephemeris Angular Momentum Limits (nominal and max excursion)
Quaternion normalization outlier threshold (max difference from 1)
Level 0R Data Directory and File Root Name
Relevant Level 0R spacecraft ancillary data contents:
S/C time-tagged inertial to body quaternion estimate
S/C time-tagged ephemeris estimate
SIRU sampling delay (latency) estimate (see note #10)
SIRU clock synchronization times – S/C clock
SIRU clock synchronization times – SIRU clock
SIRU time-tagged delta-angles
SIRU status flags
Output Preprocessed Ancillary Data File Name
Acquisition Type (Earth, Lunar, Stellar, Cal) (optional, defaults to Earth)
Trending on/off switch
WRS Path/Row (for trending)
Geometric Work Order Common Characterization ID (for trending)
Work Order ID (for trending)

7.1.4.4 Outputs

The ancillary data preprocessing algorithm outputs are shown in the following table. It is important to note that the algorithm outputs are independent of the details of SIRU/attitude data processing. Nor would the outputs change in the event that any contingency definitive attitude and/or ephemeris capabilities are required. The ability to provide a stable interface at the output of this algorithm is a large part of the motivation for separating out these ancillary data validation and conversion preprocessing operations from the model creation logic.

Preprocessed Ancillary Data
Attitude Data
Attitude data UTC epoch: Year, Day of Year, Seconds of Day
Time from epoch (one per sample, nominally 50 Hz) in seconds
ECI2ACS quaternion (vector: q1, q2, q3, scalar: q4) (one per sample)
ECEF2ACS quaternion (one per sample)
Body rate estimate (roll, pitch, yaw rate) (one per sample) in radians/second
Roll, pitch, yaw estimate (one per sample) in radians

Ephemeris Data
Ephemeris data UTC epoch: Year, Day of Year, Seconds of Day
Time from epoch (one per sample, nominally 1 Hz) in seconds
ECI position estimate (X, Y, Z) (one set per sample) in meters
ECI velocity estimate (Vx, Vy, Vz) (one set per sample) in meters/second
ECEF position estimate (X, Y, Z) (one set per sample) in meters
ECEF velocity estimate (Vx, Vy, Vz) (one set per sample) in meters/second
Trending Data
WRS Path/Row
Acquisition Date/Time
Geometric Common Characterization ID
Work Order ID
Ephemeris data start UTC time (year, day of year, seconds of day)
Number of ephemeris data points
Number of out of limit ephemeris points
Attitude data start UTC time (year, day of year, seconds of day)
Number of attitude data points
Number of out of limit attitude data points

7.1.4.5 Options

Trending On/Off Switch

7.1.4.6 Prototype Code

Input to the executable is an ODL file; output is a HDF5 formatted preprocessed ancillary data file. Status messages and sample trending outputs are written to standard output. An ancillary data ASCII log file is also created to capture messages regarding detected data problems.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The following text is a brief description of the main set of modules used within the prototype with each module listed along with a very short description. It should be noted that not all library modules are referenced in the explanations below. The modules within the main “ancillary” directory of the prototype are discussed and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

ancillary – Main procedure that retrieves the input parameters (using getpar), invokes the oli_run_preprocessing library module to perform ancillary data processing, and performs the final trending data output.

getpar – Retreives the user-provided ODL parameters.

oli_run_preprocessing – Library routine that manages the output ancillary log file (anc.log), and invokes other library routines (see below) to load and process the spacecraft ancillary data from the Level 0R input.

oli_ancillary_log_open – Opens the anc.log output file.

oli_get_header_from_l0r – Library routine that reads the L0R line headers to determine the scene start and stop times.

oli_get_ephemeris_from_l0r – Library routine that loads the ephemeris data from the input Level 0R data, quality checks the ephemeris using radius magnitude and angular momentum tests, corrects timing jitter (if any) in the ephemeris sample times, filters the ephemeris using a gravitational potential model, and creates the output ephemeris in both ECEF and ECI coordinates.

l8_correct_ephem_time – Library routine that corrects timing jitter by comparing time differences to the corresponding position differences divided by the velocity.

l8_kalman_smooth_gps – Library routine that uses a Kalman filter to smooth the ephemeris ensuring consistency with an Earth geopotential model.

geo_earth_second_partial_x – Compute gravitational acceleration for an ephemeris state vector, in the X direction.

geo_earth_second_partial_y – Compute gravitational acceleration for an ephemeris state vector, in the Y direction.

geo_earth_second_partial_z – Compute gravitational acceleration for an ephemeris state vector, in the Z direction.

geo_ec2ic – Convert ECEF to ECI.

geo_ic2ec – Convert ECI to ECEF.

exx_calc_gha – Compute Greenwich apparent sidereal time (GAST), and the corresponding precession and nutation angles at the specified time.

exx_precession – Apply the precession rotation to the input ECIJ2000 vector to convert it to ECI mean of date (ECIMOD).

exx_nutation – Apply the nutation rotation to the input ECIMOD vector to convert it to ECI true of date (ECITOD).

xxx_rotatez – Apply the GAST rotation (including UT1-UTC offset) to convert ECITOD to ECEF of date (true pole).

exx_polar_motion – Apply the polar motion correction to convert ECEF of date to WGS84/mean ECEF (mean pole).

oli_get_attitude_from_l0r – Library routine that loads the attitude quaternion and SIRU data from the Level 0R input, windows the data to ensure that it falls within the ephemeris data interval, and creates an integrated output attitude data stream in both roll-pitch-yaw and quaternion form. This unit has been modified to support processing without SIRU data if the SIRU time synchronization process (see process_siru_times below) fails.

get_siru2rpy – Construct matrix to convert 4 SIRU channel measurements to 3 orthogonal (roll-pitch-yaw) attitude measurements. Used only if SIRU processing is included.

process_siru_times – Process the SIRU clock and sync codes to create spacecraft epoch times for all SIRU samples. Used only if SIRU processing is included.

l8_iru2acs – Library routine that applies the SIRU to spacecraft alignment correction (from the CPF) to convert SIRU data to the attitude control system coordinate system. Used only if SIRU processing is included.

l8_sc_attitude – Library routine that converts ECI quaternions to roll-pitch-yaw using ephemeris data.

l8_init_iru – Library routine that removes orbital motion, which is no longer simply a pitch thanks to off-nadir pointing and yaw steering, from the SIRU data for Earth acquisitions, using the ECI ephemeris. Used only if SIRU processing is included.

l8_kalman_smooth_iru – Library routine that blends quaternion-derived absolute roll-pitch-yaw with SIRU-derived roll-pitch-yaw rates using a Kalman filter, to construct an integrated attitude data stream. Used only if SIRU processing is included.

l8_movesat – Library routine that Lagrange interpolates the ephemeris data at the desired time.

geo_eci2orb – Library routine that constructs the ECI-to-orbital rotation matrix from an ephemeris state vector.

euler2quat – Converts a rotation matrix to a quaternion.

oli_ancillary_log_close – Closes the anc.log output file.

trend_anc_to_database – Dummy routine to write the collected trending data to standard output. This would be replaced by a database call in the operational implementation.

7.1.4.7 Procedure

The primary tasks performed by the ancillary data preprocessing algorithm are to:

7. Preprocess the ancillary ephemeris data:
    a. Load the spacecraft ephemeris data from the interval ancillary data stream.
    b. Validate the ephemeris points using orbital radius and angular momentum thresholds.
    c. Convert the ephemeris time codes from spacecraft time to a UTC time epoch at the first ephemeris data record time.
    d. Correct any time jitter in the ephemeris data samples.
    e. Repair any bad ephemeris points by interpolation/propagation.

f. Perform a coordinate conversion to provide the ephemeris in both Earth Centered Earth Fixed (ECEF) and Earth Centered Inertial (ECI) of epoch J2000.0 coordinates.
    i. Convert the incoming ECEF ephemeris state vectors to ECI J2000.
    ii. Convert the ECI J2000 state vectors back to ECEF, removing the effects of Earth rotation from the velocity vectors, as described below.

8. Preprocess the ancillary attitude data:
    a. Load the spacecraft attitude data from the interval ancillary data stream.
    b. Validate the quaternion estimates by computing the magnitude of each and comparing it to 1.
    c. Window the attitude data to ensure that the attitude data are completely within the ephemeris data interval.
    d. Convert the attitude time codes from spacecraft time to a UTC time epoch at the first attitude data record time.
    e. Process the raw SIRU data time stamps to compute sample times relative to the spacecraft clock (if SIRU processing required). SIRU processing is suppressed if this process fails due to the lack of a valid SIRU time synchronization event in the ancillary data interval.
    f. Convert the raw SIRU integrated angle counts to angular rates (if SIRU processing required and not suppressed).
    g. Rotate the SIRU data to the ACS coordinate system (if SIRU processing required and not suppressed).
    h. Correct the SIRU data for the effects of orbital motion (Earth-view images only, lunar and stellar acquisitions are left in ECI). Only performed if SIRU processing is required and not suppressed.
    i. Convert the quaternions to roll, pitch and yaw using the ECI ephemeris data.
    j. Filter the SIRU and quaternion-derived roll, pitch, and yaw values to generate an integrated roll, pitch, yaw and roll-rate, pitch-rate, yaw-rate attitude sequence at the full SIRU data rate. NOTE: This step will only be used if SIRU data processing is required. If SIRU data processing is not required or is suppressed, attitude estimates flagged as outliers will be replaced by linear interpolation.
    k. Convert the roll, pitch, yaw values to ECI quaternions using the ECI ephemeris data.
    l. Convert the roll, pitch, yaw values to ECEF quaternions using the ECEF ephemeris data.

9. Create the output ephemeris and attitude data set containing:
    a. Attitude Data
        i. Attitude interval UTC epoch as: Year, Day of Year, Seconds of Day.
        ii. Attitude sample time offsets from the UTC epoch (in seconds) – one per sample. There will nominally be 50 samples per second.
        iii. Body/ACS to ECI quaternions (vector part q1, q2, q3 and scalar part q4) – one set per sample.

iv. Body/ACS to ECEF quaternions (vector part q1, q2, q3 and scalar part q4) – one set per sample.

v. Body inertial rotation rates (roll rate, pitch rate, yaw rate) in radians/second – one set per sample.

vi. Roll, pitch, and yaw in radians – one set per sample.

b. Ephemeris Data

i. Ephemeris interval UTC epoch as: Year, Day of Year, Seconds of Day.

ii. Ephemeris sample time offsets from the UTC epoch (in seconds) – one per sample. There will nominally be one sample per second.

iii. ECI X, Y, Z position in meters – one set per sample.

iv. ECI X, Y, Z velocity in meters/second – one set per sample.

v. ECEF X, Y, Z position in meters – one set per sample.

vi. ECEF X, Y, Z velocity in meters/second – one set per sample. Note that these are actually ECI velocities rotated into the ECEF coordinate system, not true ECEF velocities which would include Earth rotation velocity.

Steps 1.a., 2.a. and 3 above are input/output functions and are not described further here. The remaining steps are described in greater detail in the sub-algorithms below.

Convert Spacecraft Time Code to UTC

The convert spacecraft time code to UTC is a general purpose sub-algorithm that is used by the more specific ancillary data preprocessing sub-algorithms below. Spacecraft time codes are TAI (Temps Atomique International or International Atomic Time) offsets from the J2000 epoch. Since TAI and UTC differ only by leap seconds, the conversion to UTC amounts to a leap second correction. The spacecraft (J2000) epoch UTC date/time is hard coded (in a #define statement) to prevent it being changed inadvertently. See note #6 for more explanation.

1. Load the leap second table from the CPF. The leap second table is represented as the date that each leap second since 01 January 1972 was declared.
2. Scan the leap second table and compute the number of leap seconds prior to the J2000 spacecraft epoch.
3. Scan the leap second table and compute the number of leap seconds prior to the current spacecraft date/time. This is done by converting the spacecraft time code (TAI offset from J2000) to UTC (without any leap second correction) and then using the resulting UTC date to determine the number of leap seconds.
4. Subtract the leap second total for the spacecraft J2000 epoch (result of step 2) from the leap second total for the time code (result of step 3) to compute the number of leap seconds from the spacecraft epoch to the current spacecraft time. The resulting number of leap seconds since J2000 is stored the first time it is computed and used in each subsequent time code to/from UTC conversion operation.
5. Subtract the number of leap seconds since J2000 from the current spacecraft time code.
6. Add the adjusted time code to the UTC date/time for the spacecraft J2000 epoch to yield the UTC date/time corresponding to the spacecraft time code.

ECI to/from ECEF Coordinate Transformation

The transformation from ECI of epoch J2000 (mean equator and equinox of J2000.0) to ECEF (WGS84) coordinates is a time-varying rotation due primarily to the Earth's rotation, but it also contains more slowly varying terms for precession, astronomic nutation, and polar wander. The ECI-to-ECEF rotation matrix can be expressed as a composite of these transformations:

$\mathbf{T}_{ecef/eci} = \mathbf{A\ B\ C\ D}$

A = polar motion
B = sidereal time
C = astronomic nutation
D = precession

Polar Motion
The polar wander correction performs the transformation from the Earth's true spin axis (in the Terrestrial Intermediate Reference System) to the mean pole (in the International Terrestrial Reference System, or ITRS, here taken to be identical to WGS84). The polar motion corrections are tabulated in the Calibration Parameter File. The corrections for the current day are looked up from the CPF table and applied as described in section 6.5.2 of:
Kaplan, George H., United States Naval Observatory Circular No. 179, "The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models - Explanation and Implementation", U.S. Naval Observatory, Washington, D.C., October 20, 2005. This document will henceforth be referred to as Circular 179. This transformation is implemented using the *wobble* function in the NOVAS C3.1 library provided by the Naval Observatory.

Sidereal Time
The sidereal time correction performs the transformation from the inertial true-of-date system (true equator and equinox of date) to the Earth fixed true-of-date (true pole or terrestrial intermediate reference) system. It applies the polar rotation due to Greenwich Apparent Sidereal Time (GAST) as described in Circular 179. We use the "Equinox-Based" approach described in the Circular and implemented in the *sidereal_time* function of NOVAS C3.1. Note that the sidereal time computation includes the time correction from UTC to UT1 for the current day. The "current day" would be defined by the data set UTC epoch (rather than being evaluated for each ephemeris or attitude point) to avoid the possibility of introducing leap seconds in the middle of an imaging interval. This correction is tabulated in the CPF along with the polar wander corrections.

Nutation
The nutation correction performs the transformation from the inertial mean-of-date system (mean equator and equinox of date) to the inertial true-of-date system through nutation angles. The nutation model is based on the IAU 2000 theory of nutation as described in Circular 179 and implemented in the *nutation* function of NOVAS C3.1.

Precession
The precession correction transforms the inertial of epoch J2000.0 system to the inertial mean-of-date system. The precession model is based on the IAU 2000 definition as described in Circular 179 and implemented in the *precession* function of NOVAS C3.1. Note that we do not apply the (small) frame bias correction defined in Circular 179 because our target inertial coordinate system is the inertial system of epoch J2000 (ECIJ2000.0) rather than the Geocentric Celestial Reference System (GCRS) described in the Circular.

This transformation rotates a vector from the ECI J2000.0 system to the Earth fixed (WGS84) system. For example, an ECIJ2000 position vector is converted to ECEF as follows:

$\underline{X}_{ecef} = \mathbf{T}_{ecef/eci}\ \underline{X}_{eci} = \mathbf{A\ B\ C\ D}\ \underline{X}_{eci}$

When working with ephemeris state vectors containing both position and velocity terms, there can be ambiguity in the treatment of the velocity terms when converting between Earth fixed and inertial coordinates. This ambiguity arises because the transformation is itself time varying. Taking the time derivative of the equation above yields:

$$\dot{\mathbf{X}}_{\text{ecef}} = \mathbf{T}_{ecef\ /\ eci}\dot{\mathbf{X}}_{eci} + \dot{\mathbf{T}}_{ecef\ /\ eci}\mathbf{X}_{eci}$$

The second term captures the time-varying effect of the transformation itself. The time varying effects of precession, nutation, and polar motion transformations are negligible when compared to the orbital motion of a spacecraft, but the sidereal time transformation contributes a significant effect. Keeping this in mind and expanding $\mathbf{T}_{ecef/eci}$ above yields:

$$\dot{\mathbf{X}}_{\text{ecef}} = \mathbf{ABCD}\dot{\mathbf{X}}_{eci} + \mathbf{A}\dot{\mathbf{B}}\mathbf{CD}\mathbf{X}_{eci}$$

Where the $\dot{\mathbf{B}}$ matrix is defined as:

$$\dot{\mathbf{B}} = \begin{bmatrix} -\omega^{*}\sin(GAST) & \omega^{*}\cos(GAST) & 0 \\ -\omega^{*}\cos(GAST) & -\omega^{*}\sin(GAST) & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

With: ω^{*} = Earth rotation rate in precessing reference frame
GAST = Greenwich apparent sidereal time

For useful figures and additional explanation of this transformation see: DMA TR8350.2-A, "Supplement to the Department of Defense World Geodetic System of 1984 Technical Report – Part I: Methods, Techniques, and Data Used in WGS 84 Development", Defense Mapping Agency (now National Geospatial Intelligence Agency), 1 December 1987.

This equation shows that the ECEF velocity is composed of the ECI velocity rotated into the ECEF coordinate system (the first term) plus the effect of Earth rotation (the second term). Note that Earth rotation is modeled by the rate of change of the sidereal time transformation ($\dot{\mathbf{B}}$) applied to the (ECI true-of-date) position vector.

Whether or not the second term (Earth rotation) is included in the ECI to ECEF velocity transformation depends upon the intended purpose. The original ECEF ephemeris data received from the spacecraft contains velocity estimates that include the Earth rotation effects (i.e., "true" ECEF state vectors). The Earth rotation term must therefore be taken into account when converting these state vectors to ECI J2000. This is the coordinate system conversion that is used to accomplish step 1.f.i above. For most applications within the geometric model, however, we are only interested in the velocity vector as a direction in inertial space (e.g., when using position and velocity to define the orbital coordinate system which is the attitude control system reference). In this case, we only want the first term – the inertial velocity rotated to ECEF coordinates. We therefore, rotate the ECI J2000 ephemeris state vectors back to "pseudo" ECEF coordinates without including the Earth rotation term.

To summarize, the ECI/ECEF coordinate system transformations applied to the incoming ephemeris data from the ancillary data file are:

ECEF to ECI:

$$\mathbf{X}_{\text{eci}} = \mathbf{D}^T \mathbf{C}^T \mathbf{B}^T \mathbf{A}^T \mathbf{X}_{ecef}$$

$$\dot{\mathbf{X}}_{\text{eci}} = \mathbf{D}^T \mathbf{C}^T \mathbf{B}^T \mathbf{A}^T \left(\dot{\mathbf{X}}_{ecef} - \mathbf{A}\dot{\mathbf{B}}\mathbf{C}\mathbf{D}\mathbf{X}_{eci} \right)$$

Noting that the **A**, **B**, **C**, and **D** matrices are orthogonal so that their inverses are equal to their transposes.

ECI to (pseudo) ECEF:

$$\mathbf{X}_{\text{ecef}} = \mathbf{A}\mathbf{B}\mathbf{C}\mathbf{D}\mathbf{X}_{eci}$$

$$\dot{\mathbf{X}}'_{\text{ecef}} = \mathbf{A}\mathbf{B}\mathbf{C}\mathbf{D}\dot{\mathbf{X}}_{eci}$$

Noting that the position vector is the same as the original value, but the velocity vector is not, as indicated by the prime notation.

Correct Ephemeris Sub-Algorithm

The correct ephemeris sub-algorithm performs steps 1.b., 1.c., and 1.d. above. This sub-algorithm will quality check the ephemeris data and correct any timing jitter errors in the ephemeris solution. The ephemeris values are used to calculate satellite position in the WGS84 Earth Centered Earth Fixed (ECEF) frame.

a) Extract the ephemeris data records from the ancillary data
b) Search the ancillary data ephemeris records and find the first and last valid ephemeris records in the interval. Extract the time tags for these records.
c) Set the ephemeris epoch to the time associated with the start index found in step b) converted to UTC (see Convert Spacecraft Time Code to UTC sub-algorithm above). Retain the corresponding epoch spacecraft time as it will be subtracted from the other ephemeris samples.
d) Loop on ephemeris starting at and ending at indexes found in b.
    1) Set ephemeris sample time to the time code from ancillary data minus the ephemeris start time code, i.e. convert times to offsets from the ephemeris epoch defined in c).
    2) Convert the ECEF ephemeris position and velocity vectors to ECI J2000 so that the angular momentum check, and subsequent ephemeris smoothing algorithms can operate in inertial space.
    3) Get angular momentum and orbital radius nominal values and allowable deviation thresholds from the CPF: angmo_nom, angmo_delta, orbrad_nom, orbrad_delta.
    4) Calculate orbital radius to compare against threshold:
    radius = | $\underline{p}$ |
    Where: $\underline{p}$ = ephemeris position vector
    5) Calculate angular momentum of ephemeris to compare against threshold:

$$\text{angular momentum} = \left\| \vec{p} \times \vec{v} \right\|$$

    where:
    p = satellite positional vector
    v = satellite velocity vector
    6) Check orbital radius and angular momentum against nominal values and thresholds from CPF:
    If | radius – orbrad_nom | <= orbrad_delta AND

| angular momentum – angmo_nom | <= angmo_delta
Then store the ephemeris point for processing.
Otherwise, report the bad ephemeris point as an outlier.

If fewer than 4 (the minimum required to support Lagrange interpolation) valid ephemeris points are found a fatal error is returned for Earth-view, lunar, and stellar acquisitions. For solar cal acquisitions, additional ephemeris points are propagated using the process model described in the Smooth Position and Velocity Sub-Algorithm below, until 4 points are available.

The ECI ephemeris is reinterpolated, using the following method, to remove any small time jitters that are present.

Let vx, vy, and vz be the measured velocity.
Let px, py, and pz be the measured position.

a) Loop through the ephemeris points (i = 0 to N-1) computing the distances between adjacent points:

   If first ephemeris point (i = 0) set $d_0 = 0$.

   If ephemeris point is not first value, then

   1) Calculate difference in ephemeris from point i and i-1

      $dx_i = px_i - px_{i-1}$
      $dy_i = py_i - py_{i-1}$
      $dz_i = pz_i – pz_{i-1}$

   2) Calculate magnitudes of the delta position and the velocity vectors

      $s_i = sqrt(dx_i * dx_i + dy_i * dy_i + dz_i * dz_i)$
      $sv_i = sqrt(vx_i * vx_i + vy_i * vy_i + vz_i * vz_i)$

   3) Calculate difference between the “predicted time” from the magnitudes calculated in a2 and the time measured difference between ephemeris points i+1 and i

      Set $d_i = s_i / sv_i$ – ephemeris $time_i$ + ephemeris $time_{i-1}$

b) Calculate average difference of time differences from a.

Let $avg = \frac{\sum_{i=1}^{N-1} d_i}{N-1}$

Where N = number of ephemeris points

c) Loop through the ephemeris points, adjusting times by the “predicted time difference” (a3) and the average of “predicted time difference” (b).

ephemeris time$_i$ = ephemeris time$_i$ + d$_i$ – *avg*

Using Lagrange interpolation, calculate satellite position and velocity at one second intervals, correcting any sampling timing irregularities and filling in any missing outlier points. The time-adjusted satellite position and velocity from the previous step are taken as input.

a) Loop on ephemeris values
    1) Calculate ephemeris interpolation time for current interval (multiple of one second).
    2) Convert ephemeris time from seconds to year, day of year, and seconds of day.
    3) Bracket ephemeris data for interpolation (4 valid points are needed)
        a. Use 2 points before and 2 after the interpolation time.
        b. If that would require points beyond the beginning or end of the ephemeris interval, use the first four or the last four points in the interval.
    4) Interpolate ephemeris to current time (a1) using Lagrange interpolation and bracketed values (a3).

Use the Smooth Position and Velocity sub-algorithm (see below) to smooth the ECI ephemeris. It is then converted to ECEF so that the ECI and ECEF ephemeris representations are consistent (step 1.f.ii. above).

Smooth Position and Velocity Sub-Algorithm

A Kalman smoothing filter is used to smooth the ECI position and velocity vectors to accomplish step 1.e. above. The Kalman filter assumes a random process that can be modeled as follows:

$$[X]_{k+1} = [\phi]_k [X]_k + [q]_k + [f]_k$$

where:

$[X]_K$ = (n x 1) state vector at time t_K
$[\phi]_K$ = (n x n) matrix relating X_K to X_{K+1}
$[q]_K$ = (n x 1) process noise at time t_K
$[f]_k$ = (n x 1) forcing function at time t_K

Measurements of the process are modeled as:

$$[Z]_K = [H]_K [X]_K + [v]_K$$

where:

$[Z]_K$ = (m x 1) measurement vector at time t_k
$[H]_K$ = (m x n) relates the state vector at time t_k to the measurement
$[v]_K$ = (m x 1) measurement error at time t_k

As noted below, in this application the measurements are direct observations of the state vector so n = m.

To smooth the ephemeris data the state vector X is defined as:

$$[X] = \begin{bmatrix} Xp \\ Yp \\ Zp \\ Xv \\ Yv \\ Zv \end{bmatrix}$$

where:

Xp, Yp, Zp = X, Y, Z position

Xv, Yv, Zv = X, Y, Z velocity

The measurement vector [Z] is a 6x1 vector containing the telemetry X, Y, Z positional values along with the telemetry X, Y, Z velocity values. The measurement vector looks like the state vector but contains the measured ephemeris telemetry values for time t_k whereas the state vector contains our estimate of the "true" ephemeris position and velocity values.

The discrete state transition matrix is defined as:

$$[\phi]_k = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}_k$$

where Δt is the time transition between measurement k and k+1.

The matrix [H] is defined as a 6x6 identity matrix since the measurements directly correspond to the elements of the state vector.

The forcing function, [*f*], is equal to the change in acceleration of the satellite due to the Earth's gravitational potential. The forcing function is described further in the Potential Functions sub-algorithm below.

The process noise vector $[q]_K$ represents a random forcing function that models the uncertainty in the dynamic model as a zero-mean random process with covariance [Q]. The process noise controls how strictly the filtered states will conform to the dynamic model.

The process noise covariance matrix is defined as:

$$[Q]=\begin{bmatrix} dt^2*\sigma_x{}^2+\frac{dt^4*\sigma_{xv}{}^2}{4} & 0 & 0 & \frac{dt^4*\sigma_{xv}{}^2}{4} & 0 & 0 \\ 0 & dt^2*\sigma_y{}^2+\frac{dt^4*\sigma_{yv}{}^2}{4} & 0 & 0 & \frac{dt^4*\sigma_{yv}{}^2}{4} & 0 \\ 0 & 0 & dt^2*\sigma_z{}^2+\frac{dt^4*\sigma_{zv}{}^2}{4} & 0 & 0 & \frac{dt^4*\sigma_{zv}{}^2}{4} \\ \frac{dt^3\sigma_{xv}{}^2}{2} & 0 & 0 & dt^2*\sigma_{xv}{}^2 & 0 & 0 \\ 0 & \frac{dt^3\sigma_{yv}{}^2}{2} & 0 & 0 & dt^2*\sigma_{yv}{}^2 & 0 \\ 0 & 0 & \frac{dt^3\sigma_{zv}{}^2}{2} & 0 & 0 & dt^2*\sigma_{zv}{}^2 \end{bmatrix}$$

where:

σ_x=standard deviation in X positional element/value
σ_y=standard deviation in Y positional element/value
σ_z=standard deviation in Z positional element/value
σ_{xv}=standard deviation in X velocity element/value
σ_{yv}=standard deviation in Y velocity element/value
σ_{zv}=standard deviation in Z velocity element/value

The measurement error matrix is 6x6 diagonal matrix:

$$[R]=\begin{bmatrix} m_p & 0 & 0 & 0 & 0 & 0 \\ 0 & m_p & 0 & 0 & 0 & 0 \\ 0 & 0 & m_p & 0 & 0 & 0 \\ 0 & 0 & 0 & m_v & 0 & 0 \\ 0 & 0 & 0 & 0 & m_v & 0 \\ 0 & 0 & 0 & 0 & 0 & m_v \end{bmatrix}$$

where:

m_p = variance of error in positional measurement
m_v = variance of error in velocity measurement

The Kalman filter is used to produce a set of filtered and predicted state vectors, along with estimated and predicted covariance state error matrices. These values are then used to produce a smoothed state vector. This smoothed state vector will represent the smoothed position and velocity ephemeris data.

Prediction equations:

$$[X]_K^p=[\phi]_{K-1}[X]_{K-1}+[f]_{k-1}$$
$$[P]_K^p=[\phi]_{K-1}[P]_{K-1}[\phi]_{K-1}^T+[Q]_{K-1}$$

Filter equations:

$$[K]_K = [P]_K^P [H]_K^T ([H]_K [P]_K^P [H]_K^T + [R]_K)^{-1}$$
$$[X]_K = [X]_K^p + [K]_K ([Z]_K - [H]_K [X]_K^p)$$
$$[P]_K = ([I] - [K]_K [H]_K)[P]_K^p$$

where:

[*I*] is the identity matrix
[*P*] is the error covariance matrix
[*Q*] = E[$q_t q_t$]
[R] = E[$v_t v_t$]
$[X]^P_K$ = estimate of [*X*] given measurements through t_K
$[P]^P_K$ = error covariance associated with estimate $[X]_K$
$[X]_K$ = filtered estimate at t_K
$[P]_K$ = filtered estimate at t_K

Note that the filtering step is skipped for points flagged as outliers so that:

$[X]_K = [X]^P_K$
$[P]_K = [P]^P_K$

Using the definitions above a new notation can be written:

$[X]_{K|K-1} = [X]^P_K$
$[P]_{K|K-1} = [P]^P_K$
$[X]_{K|K} = [X]_K$
$[P]_{K|K} = [P]_K$

The smoothing equations are then:

For n=number of points-1,…,0

$$[X]_{K|N} = [X]_{K|K} + [A]_K ([X]_{K+1|N} - [X]_{K+1|K})$$
$$[A]_K = [P]_{K|K} [\phi]^T_{K+1,K} [P]^{-1}_{K+1|K}$$
$$[P]_{K|N} = [P]_{K|K} + [A]_K ([P]_{K+1|N} - [P]_{K+1|K})[A]^T_K$$

The Kalman filter is initialized with a state vector:

$$[X]_0 = \begin{bmatrix} Px(0) \\ Py(0) \\ Pz(0) \\ Vx(0) \\ Vy(0) \\ Vz(0) \end{bmatrix}$$

where:

Px(0) = first available X positional value
Py(0) = first available Y positional value
Pz(0) = first available Z positional value
Vx(0) = first available X velocity value
Vy(0) = first available Y velocity value
Vz(0) = first available Z velocity value

The initial error covariance matrix is defined as:

$$[P]_0 = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_z^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{xv}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{yv}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{zv}^2 \end{bmatrix}$$

where:

σ_x = initial standard deviation in X position
σ_y = initial standard deviation in Y position
σ_z = initial standard deviation in Z position
σ_{xv} = initial standard deviation in X velocity
σ_{yv} = initial standard deviation in Y velocity
σ_{zv} = initial standard deviation in Z velocity

- Initialize the state vector, error covariance matrix, and measurement error matrix
- Loop on ephemeris points
    - Calculate Δt (time difference between sample time i+1 and i)
    - Calculate process noise matrix
    - Calculate Kalman gain
    - Filter state vector and error covariance matrix
    - Predict error covariance error matrix
    - Calculate force (acceleration) of Earths mass
    - Predict state
- Initialize Δt to nominal delta time (1 sec)
- Loop on ephemeris (reverse order for smoothing)
    - Calculate smoothed gain
    - Calculate smoothed state
    - Calculate Δt (time difference between sample time i+1 and i)

The resulting $[X]_{K|N}$ are the smoothed ephemeris state vectors.

7.1.4.7.1 Gravitational Potential Functions

This sub-algorithm calculates the gravitational potential of the Earth represented as acceleration (x,y,z). One way to model the Earth's gravitational potential is by:

$$\varphi = \frac{\mu}{r}\left[1 - \sum_{n=2}^{\infty} J_n \frac{r_e}{r} P_n \sin(L)\right]$$

where:

J_n = Spherical Harmonics determined by experimentation
μ = Earth's gravitational parameter
r_e = equatorial radius of Earth
P_n = Legendre Polynomials
L = geocentric latitude
sin(L) = z/r

Taking the partial derivatives of the potential function with respect to x, y, and z gives the forcing functions needed for each axis.

$$\frac{\delta\varphi}{\delta x} = \left(-\frac{\mu x}{r^3}\right)\left(1 - 3\frac{J_2}{2}\left(\frac{r_e}{r}\right)^2\left(\frac{5z^2}{r^2} - 1\right) + 5\frac{J_3}{2}\left(\frac{r_e}{r}\right)^3\left(\frac{-7z^3}{r^3} + \frac{3z}{r}\right) - 5\frac{J_4}{8}\left(\frac{r_e}{r}\right)^4\left(\frac{63z^4}{r^4} - \frac{42z}{r^r} + 3\right)\right.$$
$$- 3\frac{J_5}{8}\left(\frac{r_e}{r}\right)^5\left(\frac{231z^5}{r^5} - \frac{210z^3}{r^3} + \frac{35z}{r}\right) + \frac{J_6}{16}\left(\frac{r_e}{r}\right)^6\left(\frac{-3003z^6}{r^6} + \frac{3465z^4}{r^4} - \frac{945z^2}{r^2} + 35\right)\Big)$$
$$\left(-\frac{3\mu r_e^2}{r^4}\right)\left(C_{21}\frac{z}{r} - 5C_{21}\frac{x^2 z}{r^3} - 5S_{21}\frac{xyz}{r^3} + 2C_{21}\frac{x}{r} - 5C_{22}\frac{x\left(x^2+y^2\right)}{r^3} + 2S_{22}\frac{y}{r} - 10S_{22}\frac{x^2 y}{r^3}\right)$$

$$\frac{\delta\varphi}{\delta y} = \left(-\frac{\mu y}{r^3}\right)\left(1 - 3\frac{J_2}{2}\left(\frac{r_e}{r}\right)^2\left(\frac{5z^2}{r^2} - 1\right) + 5\frac{J_3}{2}\left(\frac{r_e}{r}\right)^3\left(\frac{-7z^3}{r^3} + \frac{3z}{r}\right) - 5\frac{J_4}{8}\left(\frac{r_e}{r}\right)^4\left(\frac{63z^4}{r^4} - \frac{42z}{r^r} + 3\right)\right.$$
$$- 3\frac{J_5}{8}\left(\frac{r_e}{r}\right)^5\left(\frac{231z^5}{r^5} - \frac{210z^3}{r^3} + \frac{35z}{r}\right) + \frac{J_6}{16}\left(\frac{r_e}{r}\right)^6\left(\frac{-3003z^6}{r^6} + \frac{3465z^4}{r^4} - \frac{945z^2}{r^2} + 35\right)\Big)$$
$$\left(-\frac{3\mu r_e^2}{r^4}\right)\left(-5C_{21}\frac{xyz}{r^3} + S_{21}\frac{z}{r} - 5S_{21}\frac{y^2 z}{r^3} - 2C_{22}\frac{y}{r} - 5C_{22}\frac{y\left(x^2+y^2\right)}{r^3} + 2S_{22}\frac{x}{r} - 10S_{22}\frac{y^2 x}{r^3}\right)$$

$$\frac{\delta\varphi}{\delta z} = \left(-\frac{\mu z}{r^3}\right)\left(1 + 3\frac{J_2}{2}\left(\frac{r_e}{r}\right)^2\left(3 - \frac{5z^2}{r^2}\right) + \frac{J_3}{2}\left(\frac{r_e}{r}\right)^3\left(\frac{30z}{r} - \frac{35z^3}{r^3} - \frac{3r}{z}\right)\right.$$
$$- \frac{J_4}{8}\left(\frac{r_e}{r}\right)^4\left(\frac{-70z^2}{r^2} + \frac{63z^4}{r^4} + 15\right) - \frac{J_5}{8}\left(\frac{r_e}{r}\right)^5\left(\frac{-945z^3}{r^3} + \frac{693z^5}{r^5} + \frac{315z}{r} - \frac{15r}{z}\right)$$
$$\left. + \frac{J_6}{16}\left(\frac{r_e}{r}\right)^6\left(\frac{4851z^4}{r^4} - \frac{3003z^6}{r^6} - \frac{2205z^2}{r^2} + 245\right)\right)$$

$$\left(-\frac{3\mu r_e^2}{r^4}\right)\left(C_{21}\frac{x}{r}-5C_{21}\frac{xz^2}{r^3}+S_{21}\frac{y}{r}-5S_{21}\frac{yz^2}{r^3}-5C_{22}\frac{z\left(x^2-y^2\right)}{r^3}-10S_{22}\frac{xyz}{r^3}\right)$$

The heritage implementation uses the following six functions to invoke the potential calculations:

p1x - first derivative of X (velocity)
p1y - first derivative of Y (velocity)
p1z - first derivative of Z (velocity)
p2x - second derivative of X (acceleration)
p2y - second derivative of Y (acceleration)
p2z - second derivative of Z (acceleration)

These functions are used to populate the six elements of the forcing function used to Kalman smooth the ephemeris data.

Attitude Data Preprocessing

This sub-algorithm validates the quaternion attitude estimates and converts their spacecraft time codes to accomplish steps 2.b., 2.c., and 2.d. above.

a) Extract the attitude quaternion data records from the ancillary data
b) Search the ancillary data attitude records and find the first and last valid attitude records in the interval. Extract the time tags for these records.
c) Adjust the attitude data window as necessary to ensure that it fits entirely within the ephemeris data window.
d) Set the start and stop indexes for the attitude to be stored in the model to the indexes found in c).
e) Set the attitude epoch to the time associated with the start index found in step c) converted to UTC (see Convert Spacecraft Time Code to UTC sub-algorithm above). Retain the corresponding epoch spacecraft time as it will be subtracted from the other attitude samples.
f) Loop on attitude records starting at and ending at indexes found in c).
    1) Set attitude sample time to the spacecraft time code minus the attitude start time, i.e. convert times to offsets from attitude epoch.
    2) Compute the magnitude of the attitude quaternion:
    Mag = sqrt(q1*q1 + q2*q2 + q3*q3 + q4*q4)
    3) Check the magnitude against the nominal value of 1:
        a. If the magnitude is between 1-ε and 1+ε then store the value for processing. The quaternion normalization tolerance value, ε, is nominally 1e-06 (1 part per million) and stored in the CPF.
        b. If the magnitude is outside the allowable range then flag the value as an outlier.

If SIRU processing is required:

g) Extract the IMU (SIRU) data records from the ancillary data.
h) Process the SIRU clock data to construct spacecraft time codes for each SIRU sample. This is step 2.e. above and is described in the “Process SIRU Time” sub-algorithm below. If this step fails all subsequent SIRU processing is suppressed by setting a “SIRU_Valid” flag to FALSE.
i) Examine the SIRU status words, flagging any invalid points as outliers.
j) Convert the SIRU counts to angular rates. This is step 2.f. above and is described in the “Process SIRU Counts” sub-algorithm below.

k) Repeat steps b) through e) above for the SIRU data.

l) Use the SIRU epoch as the combined attitude data time epoch. In the event that the SIRU data are not used, the attitude quaternion times are used instead.

Process SIRU Time Sub-Algorithm

This sub-algorithm analyzes the SIRU clock readouts accompanying each SIRU data sample and uses these in conjunction with the SIRU clock sync offset values and SIRU sync spacecraft time codes included in the Level 0R IMU data records to construct spacecraft time codes for each SIRU data sample.

SIRU sample timing is driven by a clock internal to the SIRU unit. This SIRU clock is a 16-bit counter that increments every 4 microseconds, and rolls over when the 16-bit counter reaches its 64K limit. The SIRU clock is periodically (every 10 seconds or so) synchronized with the spacecraft clock when flight software sends a reset command. Flight software records the spacecraft time code associated with this reset and the SIRU records the offset between the clock counter value at the time of the reset and the clock counter value at the time of the current data sample in its clock sync field. These offsets are scaled to units of 1/3 of a microsecond (1/12 of the SIRU clock resolution). The clock counter value is not changed by the reset operation, so successfully locating and processing a single reset event is sufficient to establish the timing relationship between the spacecraft and SIRU clocks. The spacecraft time code associated with the reset is captured in the ancillary data as is the SIRU sync field. Several additional considerations regarding the SIRU timing data include:

1. The SIRU sync field is only populated during the 100 Hz cycle while it is being updated (i.e., while the SIRU is being resynchronized). Otherwise, this field will contain fill. A fill value of -32768 is used for this purpose.
2. One implication of the 100 Hz SIRU cycle is that some sync values will go unrecorded by the 50 Hz LDCM IMU telemetry. The syncs will be timed such that alternate values will be sampled by and therefore present in the ancillary data. This raises the question of how these unrecorded sync values will be detected and recovered. The SIRU sync spacecraft time code value that accompanies each IMU record will change for the record containing the sync. That would make it possible to use the SIRU sample clock data to recover the missing sync values, though it is probably not important to do so. The sync events that are represented in the data should be sufficient to establish the SIRU/spacecraft timing relationship.
3. Ancillary test data acquired during spacecraft comprehensive performance tests, demonstrated that the SIRU clock is not perfectly synchronized to the spacecraft clock. This was manifested as timing jitter in both the SIRU and attitude estimates (which apparently take their times from the corresponding SIRU samples) in which adjacent samples, nominally separated by 0.02 seconds, are sometimes 0.01 and sometimes 0.03 seconds apart, indicating timing drift across the 100 Hz sampling sequence. This has several effects:
    a. The SIRU rates must be computed using the actual time differences rather than the nominal time difference.
    b. The assignment of times to samples prior to the first SIRU clock sync must use the actual SIRU clock values, not the nominal timing offset.
    c. The SIRU clock syncs are not always visible every 20 seconds (1000 samples apart). The separation is sometimes 500 samples and sometimes 1500 samples due to 100 Hz sampling cycle slippage. This (partly) motivated the inclusion of logic to validate SIRU clock sync events against the previously established timing, to ensure that timing gaps are not introduced into the SIRU data. It also means that intervals shorter than 30 seconds may not contain any valid SIRU clock sync events, leading to the failure of this

algorithm and the suppression of further SIRU processing for the interval. This is unlikely in normal (Earth-view, lunar, and stellar) acquisitions but likely in solar calibration data.

4. The scaling and use of the SIRU latency estimate telemetry in the spacecraft ancillary data stream is not entirely clear. Ancillary data sets from the spacecraft comprehensive performance test (CPT) indicate that the latency is the time offset, in seconds, between the SIRU (and 50 Hz quaternion) data and the flight software 1 Hz cycle times (i.e., the times at which ephemeris and attitude filter outputs are generated). Since this offset is reflected in the time codes that accompany these data elements, the latency estimates are somewhat redundant. The baseline algorithm does not apply a latency correction.

Three items of SIRU timing telemetry will be used in the following algorithm: the SIRU clock value at the sample time (one per SIRU sample), the SIRU sync reference field (one per sample time, but only valid during resynchronization cycles), and the SIRU sync spacecraft time code (one per IMU record). These will be referred to as clock value (clock), clock sync (sync), and sync time code (time) respectively.

For each IMU record:

Compute the spacecraft time of last sync from the time code seconds and microseconds: time = seconds + microseconds/1e6

For each SIRU sample (i):

If the clock sync field is not fill:

a. Record the current sample clock and time (above) values as base_clock and base_time. Set base_sync equal to base_clock.
b. Compute the SIRU sync offset from the SIRU sync word using the 1/3 microsecond per count scaling factor. The sync word scaling is represented as a ratio relative to the SIRU clock scaling (12 sync counts per clock count):
   sync_time = SIRU sync word * siru_time_scale / SIRU sync ratio
c. Add the SIRU sync offset to the base_time. This the time (in seconds from spacecraft epoch) corresponding to the base_clock SIRU clock value.
d. Initialize the current offset, 16-bit rollover counter and previous offset value:
   siru_offset = 0
   excess_offset = 0
   last_offset = siru_offset
e. Compute the time of the current sample:
   gtime[i] = base_time + siru_offset * siru_time_scale (from CPF)
f. If this is the first valid (non-fill) value calculate all previous times by working back through the previous SIRU clock samples, subtracting each from the previous time:
   for (j = i to 1)
      Δclock = MOD(siru_clock[j]–siru_clock[j-1]+64K, 64K)
      gtime[j-1] = gtime[j] – Δclock*siru_time_scale
g. Make sure the new clock sync is consistent with previous time codes:
   If abs(gtime[i]-gtime[i-1]-0.02) > 0.02
      gtime[i] = gtime[i-1] + 0.02
      base_time = gtime[i] – siru_offset*siru_time_scale
   This is a coarse test that ensures that the sync update does not introduce a timing adjustment of more than a full 0.02 second sample time. A warning message is generated if this adjustment is made. This test ensures that the SIRU

time codes are consistent and, at a minimum, are based on the first sync time in the interval.

Otherwise:

If no valid sync fields have been found go to the next point

Otherwise:

Note: All arithmetic involving clock and sync variables is modulo 64K.

Check for a sync that was not sampled:

i. If time > base_time + 0.02 (a resync occurred) AND
time < gtime[i-1] + 0.02 (it occurred before this sample):
   1. Reconstruct the sync time:
      sync_time = gtime[i-1] + 0.02 - time
   2. Update the base_time:
      base_time = time + sync_time
   3. Reset the other sync cycle variables:
      base_clock = clock
      base_sync = clock
      siru_offset = 0
      excess_offset = 0

ii. Otherwise:
   1. Calculate the sample time offset based on the current sync variables:
      siru_offset = clock – base_sync (modulo 64K)
   2. Correct for previous 16-bit rollover:
      siru_offset += excess_offset
   3. See if 16-bit rollover occurred on this sample and increment rollover and offset variables if so:
      if (siru_offset < last_offset)
         excess_offset += 0x010000
         siru_offset += 0x010000

iii. Compute the time of the current sample:
gtime[i] = base_time + siru_offset * siru_time_scale

iv. Set the last offset value to the current value (used to detect missed sync resets):
last_offset = siru_offset

If no valid sync values were detected return an error.

This procedure performs step 2.e. above.

Process SIRU Counts Sub-Algorithm

This sub-algorithm converts the raw SIRU data counts to angular rates.

For each SIRU sample:

1) If the sample's SIRU validity flags are not set
   a. Mark the point as an outlier.
   b. If this is the first point, set the angular rates to zero.
   c. Otherwise, set the angular rates to the previous sample values.
2) For valid SIRU samples:
   a. If this is not the first point, compute the difference between the current integrated angle reading and the previous reading for each of the 4 SIRU axes.

b. If this is the first point, compute the difference between the next integrated angle reading and the current reading for each of the 4 SIRU axes. If the next point is invalid, mark the current point as an outlier and set the angular rates to zero.
c. Check for SIRU reset/rollover on each axis:
    i. If the value of the angle difference is > 32K, subtract 64K
    ii. If the value of the angle difference is < -32K, add 64K
d. Scale the counts to radians using the SIRU scale factor from the CPF.
e. The SIRU delta angle measurements are converted to rates by dividing by the delta time computed from the SIRU sample time codes. This could also be done after the four SIRU axis measurements are converted to roll-pitch-yaw measurements using the method described in the next section.

This procedure performs step 2.f. above. For lunar and stellar intervals, all SIRU samples are flagged as outliers so that they will be deweighted by the attitude Kalman filter.

Rotate SIRU Sub-Algorithm
This sub-algorithm rotates the SIRU data to the attitude control system (ACS) coordinate frame to accomplish step 2.g. above. Note that this sub-algorithm will only be used if SIRU data processing is required.

Construct the roll, pitch, and yaw rotational matrices from SIRU measurements and rotate angles to the ACS coordinate system. Note that rather than reporting roll, pitch, and yaw rotations directly, the SIRU reports rotations about four non-orthogonal axes oriented in an octahedral tetrad. These four correlated measurements must first be reduced to rotations about the three orthogonal X-Y-Z axes using the SIRU axis vectors from the CPF. These vectors define the orientations of the four SIRU axes, and are nominally:

SIRU A: [+0.57735027 +0.57735027 +0.57735027]
SIRU B: [+0.57735027 -0.57735027 +0.57735027]
SIRU C: [-0.57735027 -0.57735027 +0.57735027]
SIRU D: [-0.57735027 +0.57735027 +0.57735027]

Any one of these SIRU axes can be lost and the rotations about the X-Y-Z axes can still be recovered. The vector for a failed SIRU axis should be set to zero.

Convert SIRU angles to roll-pitch-yaw:

Construct the [S] matrix where the columns of this 3 by 4 matrix contain the four SIRU axis vectors:

$$[S] = \begin{bmatrix} SIRUAx & SIRUBx & SIRUCx & SIRUDx \\ SIRUAy & SIRUBy & SIRUCy & SIRUDy \\ SIRUAz & SIRUBz & SIRUCz & SIRUDz \end{bmatrix}$$

Construct the 3 by 4 SIRU to roll-pitch-yaw conversion matrix:

$$[SIRU2RPY] = \left([S][S]^T\right)^{-1}[S]$$

Construct the 4 by 1 SIRU observation vector:

$$[SIRUOBS] = \begin{bmatrix} \tan(\frac{\theta_A}{2}) \\ \tan(\frac{\theta_B}{2}) \\ \tan(\frac{\theta_C}{2}) \\ \tan(\frac{\theta_D}{2}) \end{bmatrix}$$

where, θ_n = angle for SIRU axis n

Convert the four SIRU observations to three roll-pitch-yaw angles in the SIRU coordinate system:

$$[RPYOBS] = \begin{bmatrix} \tan(\frac{roll}{2}) \\ \tan(\frac{pitch}{2}) \\ \tan(\frac{yaw}{2}) \end{bmatrix} = [SIRU\,2RPY][SIRUOBS]$$

$$[RPYANG] = \begin{bmatrix} roll \\ pitch \\ yaw \end{bmatrix} = \begin{bmatrix} 2 * a\tan(RPYOBS_1) \\ 2 * a\tan(RPYOBS_2) \\ 2 * a\tan(RPYOBS_3) \end{bmatrix}$$

Construct the perturbation matrix using the SIRU roll-pitch-yaw angles:

$$[\text{perturbation}] = [IRU\ to\ ACS][\text{yaw}][\text{pitch}][\text{roll}][\text{ACS to IRU}]$$

Where [IRU to ACS] is the SIRU to attitude matrix found in the CPF and [ACS to IRU] is its inverse.

The SIRU measured roll, pitch and yaw in ACS coordinates are then:

$$\text{roll} = -\tan^{-1}(\text{perturbation}_{2,1} / \text{perturbation}_{2,2})$$
$$\text{pitch} = \sin^{-1}(\text{perturbation}_{2,0})$$
$$\text{yaw} = -\tan^{-1}(\text{perturbation}_{1,0} / \text{perturbation}_{0,0})$$

7.1.4.7.2 Correct for Orbital Motion in SIRU Data Sub-Algorithm

The spacecraft SIRU senses rotations relative to inertial space so it will measure the orbital pitch used to maintain spacecraft pointing as well as any deviations from that nominal alignment with the orbital coordinate system. In using the SIRU data to densify or repair the spacecraft attitude estimates they are blended in the orbital coordinate system. Before this can be done it is necessary to correct the SIRU data for the time-varying orientation of the orbital frame relative to inertial space. This is step 2.h. above.

To account for off-nadir viewing and yaw steering effects a reference attitude vector representing the mean roll-pitch-yaw for the scene is also provided. This is necessary because the orbital pitch effect will show up in more than just the spacecraft pitch axis if the spacecraft body is not aligned with the orbital coordinate system. The reference attitude is calculated as the average of the roll-pitch-yaw values derived from the quaternion data for the attitude interval.

Note that this sub-algorithm will only be used if SIRU data processing is required.

If the acquisition type is Earth-viewing then:

Loop on all SIRU values:

1) Calculate the ECI position and velocity of satellite at time t_0 using Lagrange interpolation. The l8_movesat unit does this. That sub-algorithm is included in the LOS Model Creation algorithm.

2) Calculate the transformation matrix from the satellite orbit system to the spacecraft body/ACS coordinate system (ORB2ACS) using the input reference mean attitude. This is the transpose of the ACS2ORB matrix shown in the “Convert Roll, Pitch, Yaw to Quaternion” section below.

3) Calculate the transformation matrix from ECI to satellite orbit system for time t_0 and t_n (the inverse of the ORB2ECI matrix presented in the next section).

Using the satellite position and velocity at times t_0 and t_n, the following matrix transformations can be calculated:

$$[\text{eci2orb}]_{time=t0} \Rightarrow [\text{eci2orb}]_{t0}$$
$$[\text{eci2orb}]_{time=tn} \Rightarrow [\text{eci2orb}]_{tn}$$

Calculate the transformation from the Orbit system to ECI for time t_n using the ephemeris state vector at time t_n.

4) Use the ORB2ACS matrix to compute the ECI2ACS matrices from the ECI2ORB matrices:

$[\text{eci2acs}]_{t0} = [\text{orb2acs}]\ [\text{eci2orb}]_{t0}$
$[\text{eci2acs}]_{tn} = [\text{orb2acs}]\ [\text{eci2orb}]_{tn}$

Since the eci2acs matrix is orthogonal, acs2eci can be calculated as:

$$[\text{acs2eci}] = [\text{eci2acs}]^T$$

6) Calculate the amount of roll, pitch, and yaw due to the satellite’s orbit.

The roll, pitch, and yaw due to the orbital motion of the satellite can be found by looking at the matrix transformation from spacecraft frame reference at time t_n to spacecraft frame reference t_0.

$$[\text{acs}_n 2\text{acs}_0] = [\text{eci2acs}_0][\text{acs}_n 2\text{eci}]$$

$$\Delta\text{roll} = -\frac{\tan^{-1}\left(\frac{(\text{acs}_n 2\text{acs}_0)_{2,1}}{(\text{acs}_n 2\text{acs}_0)_{2,2}}\right)}{\Delta\text{time}}$$

$$\Delta\text{pitch} = \frac{\sin^{-1}\left((\text{acs}_n 2\text{acs}_0)_{2,0}\right)}{\Delta\text{time}}$$

$$\Delta\text{yaw} = -\frac{\tan^{-1}\left(\frac{(\text{acs}_n 2\text{acs}_0)_{1,0}}{(\text{acs}_n 2\text{acs}_0)_{0,0}}\right)}{\Delta\text{time}}$$

$\Delta\text{time} = t_n - t_0$

This formulation computes the orbital attitude rate correction and assumes that the SIRU data are, or have been converted to, rates.

5) Remove orbital motion attitude delta from original values.

$$\text{roll} = -\text{roll} - \Delta\text{roll}$$
$$\text{pitch} = -\text{pitch} - \Delta\text{pitch}$$
$$\text{yaw} = -\text{yaw} - \Delta\text{yaw}$$

The sign is swapped to convert the SIRU angles/rates from body-to-orbit to orbit-to-body.

7.1.4.7.3 Convert to Spacecraft Roll, Pitch, and Yaw Sub-Algorithm

The attitude data is given as quaternions in the ECI reference frame (ECI2ACS). The quaternions are converted to roll, pitch, and yaw values in the ACS reference frame per step 2.i. above.

We first take the conjugate of the incoming ECI2ACS quaternion (**q**) to calculate the corresponding ACS2ECI quaternion (**q'**).

$q'_1 = -q_1$
$q'_2 = -q_2$
$q'_3 = -q_3$
$q'_4 = q_4$

The direction cosines (transformation) matrix from the ACS reference axis to the ECI reference system (ACS2ECI) is constructed from the ACS2ECI quaternion, **q'**, as:

ACS2ECI =

$$\begin{bmatrix} q'_1{}^2-q'_2{}^2-q'_3{}^2+q'_4{}^2 & 2(q'_1 q'_2+q'_3 q'_4) & 2(q'_1 q'_3-q'_2 q'_4) \\ 2(q'_1 q'_2-q'_3 q'_4) & -q'_1{}^2+q'_2{}^2-q'_3{}^2+q'_4{}^2 & 2(q'_2 q'_3+q'_1 q'_4) \\ 2(q'_1 q'_3+q'_2 q'_4) & 2(q'_2 q'_3-q'_1 q'_4) & -q'_1{}^2-q'_2{}^2+q'_3{}^2+q'_4{}^2 \end{bmatrix}$$

The ACS2ECI transformation matrix can also be defined as the product of the inverse of the spacecraft's attitude perturbation matrix **P** and the transformation matrix from the orbital reference system to the ECI reference system (ORB2ECI)

The relationship between the orbital and ECI coordinate systems is based on the spacecraft's instantaneous ECI position and velocity vectors. The rotation matrix to convert from orbital to ECI can be constructed by forming the orbital coordinate system axes in ECI coordinates:

$$\vec{n} = -\frac{\vec{p}}{\left\|\vec{p}\right\|}$$

$$\vec{h} = \frac{\left(\vec{n}\times\vec{v}\right)}{\left\|\vec{n}\times\vec{v}\right\|}$$

$$\vec{cv} = \vec{h}\times\vec{n}$$

$$\left[\text{ORB2ECI}\right] = \left[\vec{cv} \quad \vec{h} \quad \vec{n}\right]$$

where:

p = spacecraft position vector in ECI
v = spacecraft velocity vector in ECI
n = nadir vector direction
h = negative of angular momentum vector direction
cv = circular velocity vector direction
[ORB2ECI] = rotation matrix from orbital to ECI

The transformation from orbital to ECI coordinates is the inverse of the ECI to orbital transformation matrix. Since the ECI to orbital matrix is orthogonal the inverse is also equal to the transpose of the matrix.

$$\left[\text{ORB2ECI}\right] = \left[\text{ECI2ORB}\right]^{-1} = \left[\text{ECI2ORB}\right]^{T}$$

ACS2ECI = **[**ORB2ECI**][P $^{-1}$]**

The orbital reference system to ECI matrix must be defined at the same time as the spacecraft's attitude matrix.

The roll, pitch, and yaw values are contained in the $\mathbf{P}^{-1}$ matrix, thus:

$$\mathbf{P}^{-1} = [\text{ORB2ECI}]^{-1}[\text{ACS2ECI}]$$

The spacecraft attitude is then:

$$\text{pitch} = \sin^{-1}\left(P^{-1}_{2,0}\right)$$

$$\text{roll} = -\tan^{-1}\left(\frac{P^{-1}_{2,1}}{P^{-1}_{2,2}}\right)$$

$$\text{yaw} = -\tan^{-1}\left(\frac{P^{-1}_{1,0}}{P^{-1}_{0,0}}\right)$$

For lunar and stellar intervals, the rotation to the orbital coordinate system is not performed, so the resulting roll, pitch, and yaw values are relative to the ECI system. An additional check is performed on these roll, pitch, yaw values to make sure that there are no crossings of the +/-π radians boundary, with 2π being added or subtracted as necessary to keep the attitude sequence continuous. This check is performed on all intervals but is only necessary for lunar/stellar data.

7.1.4.7.4 Smooth Euler and SIRU Sub-Algorithm

A Kalman smoothing filter is used to combine the attitude and SIRU data into one data stream and/or replace attitude estimates flagged as outliers per step 2.j. above. Note that this sub-algorithm will only be used if SIRU data processing is required and if the SIRU data are not suppressed.

Lagrange interpolation is used to synchronize the SIRU and quaternion information at the SIRU sampling interval relative to the attitude epoch time. This is necessary because the quaternion and SIRU data sample times are not necessarily uniformly spaced in the original spacecraft ancillary data. The formulation shown here assumes that the SIRU is reporting attitude rate data rather than integrated angles. Due to the increased potential for noise in rate measurements, an additional step is required to synchronize the SIRU data. Specifically, the SIRU rate measurements are integrated to form angles, the angles are Lagrange interpolated to synchronize the times, then the interpolated angles are converted back to rates. The rate to angle integration is performed as follows (the roll, pitch, and yaw axes are each processed separately using this method):

```
SIRU_angle[0] = SIRU_rate[0]*nominal_SIRU_time
For k = 1 to NSIRU-1
        SIRU_angle[k] = SIRU_angle[k-1]
                        + SIRU_rate[k]*(SIRU_time[k] – SIRU_time[k-1])
```

Performing the time regularizing interpolation in angle space suppresses any rate noise present in the SIRU data. The interpolated angles are turned back into rates, suitable for use in the Kalman smoother, as follows:

```
For k = NSIRU-1 to 1
        SIRU_rate[k] = (SIRU_angle[k] – SIRU_angle[k-1])/nominal_SIRU_time
SIRU_rate[0] = SIRU_angle[0] / nominal_SIRU_time
```

The state vector is defined as:

$$[X] = \begin{bmatrix} \text{attitude} \\ \text{iru} \\ \text{drift} \end{bmatrix}$$

where:

attitude = smoothed attitude state

iru = attitude rate state associated with SIRU

drift = slow linear drift error in IRU

The measurement matrix [Z] is a 2x1 matrix containing the Euler and SIRU attitude data for time t_k.

$$[Z] = \begin{bmatrix} \text{epa}_k \\ \text{iru}_k \end{bmatrix}$$

where:

epa = Euler attitude value at time t_k

iru = SIRU attitude value at time t_k

The state transition matrix is defined as:

$$[\phi] = \begin{bmatrix} 1 & \text{dt} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where:

dt = sample timing of SIRU

The matrix [H] is defined as:

$$[H] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

The process noise covariance matrix is defined as:

$$[Q] = \begin{bmatrix} \frac{\text{dt}^4 * \sigma_{iru}^{\ 2}}{4} & \frac{\text{dt}^3 * \sigma_{iru}^{\ 2}}{2} & 0 \\ \frac{\text{dt}^3 * \sigma_{iru}^{\ 2}}{2} & \text{dt}^2 * \sigma_{iru}^{\ 2} & 0 \\ 0 & 0 & \text{dt}^2 * \sigma_{drift}^{\ 2} \end{bmatrix}$$

where:

σ_{iru}=standard deviation of SIRU process

σ_{drift}=standard deviation of drift process

The measurement noise covariance matrix is defined as a 2x2 diagonal matrix:

$$[R] = \begin{bmatrix} m_{euler} & 0 \\ 0 & m_{iru} \end{bmatrix}$$

where:

m_{euler} = observation standard deviation noise in Euler measurement
m_{iru} = observation standard deviation noise in SIRU measurement

Samples flagged as outliers are deweighted by multiplying the measurement standard deviation by 100 for that point.

Each axis is treated as an independent data stream. The Kalman filter is used to produce a set of filtered and predicted state vectors along with estimated and predicted covariance state error matrices. These values are then used to produce a smoothed state vector. The smoothed vector attitude will represent an overall satellite attitude, or a combination of the Euler and SIRU measurements.

The Kalman filter has an initial state vector of:

$$[X]_0 = \begin{bmatrix} \mathrm{epa}(0) \\ \mathrm{iru}(0) \\ 0 \end{bmatrix}$$

where:

epa(0) = first measured quaternion
iru(0) = first measured SIRU

The initial covariance error matrix is defined as:

$$[P]_0 = \begin{bmatrix} \sigma_{epa}{}^2 & 0 & 0 \\ 0 & \sigma_{iru}{}^2 & 0 \\ 0 & 0 & \sigma_{drift}{}^2 \end{bmatrix}$$

where:

σ_{epa} = initial standard deviation in Euler
σ_{iru} = initial standard deviation in SIRU
σ_{drift} = initial standard deviation in drift

Initialize the state vector, error covariance matrix, measurement error matrix, and dt.

Loop on attitude points

- Calculate process noise matrix
- Calculate Kalman gain
- Filter state vector and error covariance matrix

- Predict error covariance error matrix
- Predict state

Loop on attitude points (reverse order for smoothing)

- Calculate smoothed gain
- Calculate smoothed state

The Kalman filtering machinery used here is the same as described in Smooth Position and Velocity Sub-Algorithm above.

If SIRU data processing is not performed, this sub-algorithm is replaced by a simple attitude outlier replacement algorithm that replaces estimates flagged as outliers above, by linearly interpolating new roll, pitch, and yaw values from the neighboring samples.

Convert Roll, Pitch, Yaw to Quaternion
The roll pitch and yaw sequences computed above are converted to ECI quaternions and to ECEF quaternions per steps 2.k. and 2.l. above. The conversion algorithm is the same in both cases, the only difference being whether the algorithm is provided with ECI ephemeris data or ECEF ephemeris data. Also see note 7.

For each attitude sample:

a) Compute the net roll-pitch-yaw by adding the bias value.
b) Use Lagrange interpolation to compute the ephemeris position and velocity at the time of the roll, pitch, yaw attitude sample.
c) Compute the rotation matrix corresponding to the roll-pitch-yaw values:
[ACS2ORB] =

$$\begin{bmatrix} \cos(\mathrm{p})\cos(\mathrm{y}) & \sin(\mathrm{r})\sin(\mathrm{p})\cos(\mathrm{y})+\cos(\mathrm{r})\sin(\mathrm{y}) & \sin(\mathrm{r})\sin(\mathrm{y})-\cos(\mathrm{r})\sin(\mathrm{p})\cos(\mathrm{y}) \\ -\cos(\mathrm{p})\sin(\mathrm{y}) & \cos(\mathrm{r})\cos(\mathrm{y})-\sin(\mathrm{r})\sin(\mathrm{p})\sin(\mathrm{y}) & \cos(\mathrm{r})\sin(\mathrm{p})\sin(\mathrm{y})+\sin(\mathrm{r})\cos(\mathrm{y}) \\ \sin(\mathrm{p}) & -\sin(\mathrm{r})\cos(\mathrm{p}) & \cos(\mathrm{r})\cos(\mathrm{p}) \end{bmatrix}$$

d) Construct the rotation matrix to convert from orbital to ECI/ECEF by forming the orbital coordinate system axes in ECI/ECEF coordinates:

$$\vec{n} = -\frac{\vec{p}}{\left\|\vec{p}\right\|}$$

$$\vec{h} = \frac{\left(\vec{n}\times\vec{v}\right)}{\left\|\vec{n}\times\vec{v}\right\|}$$

$$\vec{cv} = \vec{h}\times\vec{n}$$

$$\left[\mathrm{ORB2EC}\right] = \left[\vec{cv} \quad \vec{h} \quad \vec{n}\right]$$

where:

p = spacecraft position vector in ECI/ECEF
v = spacecraft velocity vector in ECI/ECEF
n = nadir vector direction
h = negative of angular momentum vector direction
cv = circular velocity vector direction
[ORB2EC] = rotation matrix from orbital to ECI/ECEF

e) Compute the ACS2EC rotation matrix:

[ACS2EC] = **[**ORB2EC**][**ACS2ORB**]**

f) Construct the corresponding EC2ACS quaternion:

First, noting that the ACS2EC matrix computed above can be expressed in terms of the corresponding quaternion components as:

ACS2EC =

$$\begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix}$$

We can derive the following set of equations to compute the quaternion components from the elements of ACS2EC:

1. Compute the four quantities:
   $d_1 = 1 + ACS2EC_{11} - ACS2EC_{22} - ACS2EC_{33}$
   $d_2 = 1 - ACS2EC_{11} + ACS2EC_{22} - ACS2EC_{33}$
   $d_3 = 1 - ACS2EC_{11} - ACS2EC_{22} + ACS2EC_{33}$
   $d_4 = 1 + ACS2EC_{11} + ACS2EC_{22} + ACS2EC_{33}$

2. Find the largest of these four quantities and use the corresponding equations to compute the quaternion:

$$if\ (d_1 > MAX(d_2, d_3, d_4)$$

$$q_1 = \frac{1}{2}\sqrt{(1 + ACS2EC_{11} - ACS2EC_{22} - ACS2EC_{33})} = \frac{1}{2}\sqrt{d_1}$$

$$q_2 = \frac{1}{4q_1}(ACS2EC_{12} + ACS2EC_{21})$$

$$q_3 = \frac{1}{4q_1}(ACS2EC_{13} + ACS2EC_{31})$$

$$q_4 = \frac{1}{4q_1}(ACS2EC_{23} - ACS2EC_{32})$$

elseif $(d_2 > MAX(d_1, d_3, d_4)$

$$q_2 = \frac{1}{2}\sqrt{(1 - ACS2EC_{11} + ACS2EC_{22} - ACS2EC_{33})} = \frac{1}{2}\sqrt{d_2}$$

$$q_1 = \frac{1}{4q_2}(ACS2EC_{12} + ACS2EC_{21})$$

$$q_3 = \frac{1}{4q_2}(ACS2EC_{23} + ACS2EC_{32})$$

$$q_4 = \frac{1}{4q_2}(ACS2EC_{31} - ACS2EC_{13})$$

elseif $(d_3 \geq MAX(d_1, d_2, d_4)$

$$q_3 = \frac{1}{2}\sqrt{(1 - ACS2EC_{11} - ACS2EC_{22} + ACS2EC_{33})} = \frac{1}{2}\sqrt{d_3}$$

$$q_1 = \frac{1}{4q_3}(ACS2EC_{13} + ACS2EC_{31})$$

$$q_2 = \frac{1}{4q_3}(ACS2EC_{23} + ACS2EC_{32})$$

$$q_4 = \frac{1}{4q_3}(ACS2EC_{12} - ACS2EC_{21})$$

else

$$q_4 = \frac{1}{2}\sqrt{(1 + ACS2EC_{11} + ACS2EC_{22} + ACS2EC_{33})} = \frac{1}{2}\sqrt{d_4}$$

$$q_1 = \frac{1}{4q_4}(ACS2EC_{23} - ACS2EC_{32})$$

$$q_2 = \frac{1}{4q_4}(ACS2EC_{31} - ACS2EC_{13})$$

$$q_3 = \frac{1}{4q_4}(ACS2EC_{12} - ACS2EC_{21})$$

This method avoids the numerical danger of dividing by a small number.

We then take the conjugate of the resulting ACS2EC quaternion, **q**, to yield the output EC2ACS quaternion, **q'**:

$q'_1 = -q_1$
$q'_2 = -q_2$
$q'_3 = -q_3$
$q'_4 = q_4$

7.1.4.8 Maturity

The ancillary data preprocessing algorithm includes new features (e.g., SIRU processing) but reuses many heritage components (e.g., coordinate system transformations). Some notable modifications to the heritage logic include:

1. The inertial to Earth fixed coordinate transformation logic was upgraded to include leap seconds (table in CPF to convert from TAI) and light travel time effects (for LOS projection).
2. The Landsat and EO-1 heritage algorithm for converting between Earth-fixed and inertial coordinates performs simple rotation from the ECI to the ECEF system (or vice versa) for any input vector. This has the effect of rotating the inertial velocity vector to the ECEF frame without incorporating the Earth rotation effect in the velocity. The GPS-derived LDCM ECEF ephemeris includes Earth rotation effects. As noted in the ADD above, a new velocity conversion unit was required to implement the velocity equations shown in figure A.1 of DMA TR8350.2-A:
   Position: $\mathbf{X}_{ECEF} = [\ ABCD\ ]\ \mathbf{X}_{ECI}$
   Velocity: $\mathbf{V}_{ECEF} = [\ AB'CD\ ]\ \mathbf{X}_{ECI} + [\ ABCD\ ]\ \mathbf{V}_{ECI}$
   Where: B' is the time derivative of the B matrix.
3. The heritage ephemeris time jitter correction and Kalman smoother logic has been included in the baseline algorithm but may not be necessary as the spacecraft ephemeris should be cleaner than what we got from EO-1.
4. The heritage IAU 1980 precession and nutation models were replaced with the NOVAS C3.1 implementation of the IAU 2000 models.

7.1.4.9 Notes

Algorithm assumptions and notes, including those embedded in the text above, are:

3. The attitude and position/velocity estimates produced by the spacecraft will be sufficiently accurate to achieve LDCM geolocation accuracy requirements without definitive processing of the raw attitude sensor and/or GPS data.
4. Ancillary data for the full imaging interval with 4 seconds of extra data before and after the interval, is available to provide the required geometric support data, a CPF containing the scale factors needed to convert the ancillary data to engineering units is available, and the quality thresholds needed to detect and remove or repair outliers are provided in the CPF.
5. The spacecraft ancillary data will provide attitude estimates (in the form of ECI-to-body quaternions) at the same rate that it provides SIRU data. It remains to be seen whether the spacecraft attitude estimates embody the full SIRU bandwidth. If they are overly smoothed, then the SIRU data will be used to restore the high frequency information to the sequence of attitude estimates.
6. Spacecraft time codes will be TAI offsets from the J2000 epoch. Since TAI and UTC differ only by leap seconds, the conversion to UTC amounts to a leap second correction. The spacecraft (J2000) epoch is hard coded (in a #define statement) to prevent it from being inadvertently changed.
7. Spacecraft ephemeris data will be provided in ECEF rather than ECI coordinates.
8. Ancillary data will include ephemeris and attitude records that contain time tags/time codes (e.g., seconds and fractions of seconds) that are TAI offsets from the J2000 epoch. Note that J2000 occurred at January 1, 2000, 11:59:27.816 TAI which corresponds to January 1, 2000, 11:58:55.816 UTC (ref. Space to Ground ICD 70-P58230P Rev B). These times reflect the 32.184 second offset between TAI and TDT (the J2000 epoch reference frame) and the 32 second offset between TAI and UTC as of J2000. The TAI-UTC offset at J2000 includes the fixed 10 second TAI-UTC offset as of January 1, 1972 and the 22 accumulated leap seconds between then and J2000.

9. The baseline algorithm retains the heritage roll-pitch-yaw attitude model. At some point in the future this may be replaced by a reformulated model that uses the quaternion representation directly. The sub-algorithm that converts the roll-pitch-yaw attitude representation to a quaternion may not ultimately be used in such a quaternion-based reformulation of the attitude model, since a part of that reformulation would probably involve directly filtering/smoothing the quaternion sequence rather than working in roll-pitch-yaw coordinates. That said, having the capability, provided by this sub-algorithm, to generate a quaternion attitude data representation that is identical to the roll-pitch-yaw representation would simplify the testing of any future reformulation. Note that by including both roll-pitch-yaw and quaternion representations of the attitude data, the algorithm outputs support either approach.
10. Common mathematical algorithms (e.g., matrix and vector operations, Lagrange interpolation) that can be found in standard references (e.g., Numerical Recipes in C) are cited without being described here.
11. The spacecraft estimate of SIRU latency was a late addition to the spacecraft ancillary data stream. Based on our current understanding of its meaning, it is not needed for SIRU data processing and is not included in the SIRU processing algorithm as of this writing. Should the need for this parameter be established, it should be a straightforward adjustment to the computed SIRU sample times.
12. The terms “IMU”, “IRU”, and “SIRU” are used interchangeably in this document. Inertial measurement unit (IMU) is another name for an inertial reference unit (IRU). The space inertial reference unit (SIRU) manufactured by Northrup Grumman is the particular type of IRU used by LDCM.

7.1.5 Ground Control Point Correlation Algorithm

7.1.5.1 Background/Introduction

The ground control point (GCP) correlation algorithm applies standard image matching techniques to measure the locations of a set of GCPs, each consisting of positional information and an image chip, within a Level 1Gt OLI/TIRS image. For each measured GCP the correlation status (success or failure) and the location within the image where the GCP was expected and where it was actually measured, are reported.

The GCP correlation algorithm will be used in two different contexts in the LDCM Image Processing Element. It will function as part of the primary Level 1T (L1T) product generation flow where it provides control point measurements for use by the OLI LOS Model Correction algorithm in registering the L1T products to the GLS2000 control base. The GLS control base as used here includes control from the Landsat Image Mosaic of Antarctica (LIMA) to provide global coverage. The GCP correlation algorithm will also be used for geometric assessment as a tool for measuring the locations of validation GCPs in processed L1T imagery. These measurements will be analyzed by the Geometric Accuracy Assessment algorithm for data product characterization purposes. Digital Orthophoto Quadrangle (DOQ) will be used on geometric supersites for instrument and platform characterization and calibration.

The LDCM GCP correlation algorithm is derived from the corresponding ALI algorithm used in ALIAS. Its implementation should be very similar to the gcpcorrelate application. This is a utility algorithm that is not dependent on sensor architecture.

7.1.5.2 Dependencies

The GCP correlation algorithm assumes that ground control points are available for the ground site and that the Model Creation, LOS Projection and Gridding, and Image Resampling algorithms have been executed to create an SCA-separated terrain corrected L1G image for GCP mensuration (for the LOS model correction application). It may also operate directly on an SCA-combined L1T product image for geometric accuracy assessment purposes. In either case, the image must match the GCP image chips with respect to ground sample distance, map projection, and image orientation. As such, the band selection and resolution of the input image will depend upon the flow being executed/control source being used. For standard L1T product generation processing the GLS control points (SWIR1 band, 30m resolution) will be used whereas for characterization and calibration flows the DOQ control points (panchromatic band, 15m resolution) will be used. For L1T product geometric assessment, GLS GCPs flagged as validation points will be extracted and used. A limited set of thermal (ETM+ band 6, 60m resolution) GCPs will also be available to support contingency TIRS-only accuracy characterization operations. These thermal GCPs will use a source identifier of "TM6".

GCP Retrieval

The GCP mensuration process relies upon a control point storage, management, and retrieval infrastructure (see maturity note #1 below). Though not formally part of the GCP correlation algorithm, the availability of logic to retrieve the GCPs corresponding to a particular L1G image is a dependency of the algorithm. The Landsat 7 production system is the model for this capability and will be the source of the GLS-derived operational GCPs. The GCP retrieval logic would query the GCP repository and request GCP records based upon:

1. Geography - GCPs that fall within the latitude/longitude bounds of the L1G image being correlated. Note that GCPs meeting this criterion could come from more than one WRS path/row, particularly at high latitudes.

2. GCP Source - As noted above, operational L1T product generation would use GLS control while the characterization and calibration operations would use DOQ control. Though not shown as an input to this algorithm in the table below, which takes the pre-assembled control package created by the GCP retrieval process as an input, this GLS, DOQ or TM6 control source selection would be a work order input. Note that there is no requirement to combine GLS, DOQ and/or TM6 GCPs in a single control set.
3. GCP Type - GCPs will be flagged as "control" or "validation" points so that a subset of the available GCPs can be withheld from the image correction process to provide an independent basis for accuracy assessment (see note #4). Valid query options are: CONTROL, VALIDATION, and BOTH. The CONTROL GCP type will be requested for all cases except the geometric accuracy assessment operation which will use VALIDATION points. In the event of a correlation failure, a high priority scene may be reprocessed using the BOTH option. This could help in cases where cloud cover has limited the set of usable GCPs in a particular scene. Under this scenario if a scene was deemed as necessary for processing (high priority), for characterization, calibration, or other reasons, this scene would be processed through the IAS using the BOTH option.

The GCP retrieval process would extract the GCPs meeting the specified criteria from the GCP repository and construct a GCP data package/library for input to this algorithm. This data package would include the information presented in the Inputs section and in Table 1 below. The implementation details of how the GCP data fields and image chips are stored, managed, and packaged for delivery are not addressed in this ADD.

It is probably worth mentioning that the GCP retrieval query may return no valid GCPs. This will happen if, for example, DOQ or TM6 control is requested for an area where it does not exist, or GLS control is requested for a sea ice area or island/reef where control is not available. In this case the processing system will have to be able to gracefully handle the lack of control (e.g., treat it as a correlation failure and proceed with systematic processing). This is also outside the scope of this ADD.

For the prototype code the database retrieval is mimicked in a two step process. The first is with a perl script that searches for all GCPs available on line that fit within the images geographic extent. This script will produce an ASCII file listing each valid chip (chips within image geographic extent) and relevant projection information such as UTM zone. The second step is a C executable file that will read the ASCII file created from the perl script and place all valid chips within a local directory. These chips are then resampled if they are not projected to the same UTM zone as the image file (see maturity note #3).

7.1.5.3 Inputs

The GCP correlation algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data; including data set IDs to provide unique identifiers for data trending).

Algorithm Inputs
ODL File (implementation)
Input GCP library/package name/link
Level 1G Image file name
Band to process
Output GCP measurement file name
Calibration Parameter File name (CPF contains default values for processing

parameters)
Options and Parameters
Correlation result fit method (see note 2)
Search window size (line, sample) in pixels
Maximum allowable GCP displacement in pixels
Minimum correlation strength (dimensionless)
Image fill value to ignore in correlation
Predicted GCP location offset (line, sample) in pixels (optional) (see note #5)
GCP library/package contents: (see Table 1 for details)
Number of GCPs
For each GCP:
GCP ID
GCP type (GLS, DOQ or TM6) (new)
GCP ground position (lat/lon/proj X/proj Y/height) for each GCP
Location of control point within image chip
Chip parameters (e.g., size, ground sample distance (GSD))
Image chip (see note 1)
Level 1G image contents
Image data
Image metadata (DDR) including:
Image dimensions (number of lines and samples)
Map projection
GSD/pixel size
Scene corner coordinates

7.1.5.3.1 CPF Parameters

Parameter	Type	Description
GCP Correlation Window Size	Int	Correlation window size
GCP Minimum Correlation Peak	double	Minimum allowable correlation peak strength
GCP Maximum Displacement	double	Maximum allowable measured offsets
GCP Correlation Fit Method	Int	Correlation subpixel peak fit methodology
GCP Correlation Threshold	double	Threshold of allowable fill values in correlation window
GCP Correlation Fill Value	double	Fill value to check for in correlation window

7.1.5.4 Outputs

GCP Measurements (see Table 2 for details)
GCP ID
Nominal GCP chip line/sample
GCP ground position (lat/lon/height)
Predicted GCP image line/sample
Measured offset from predicted line/sample
Correlation success flag
Correlation coefficient (new)

7.1.5.5 Options

Correlation Fit Method (only one choice in baseline algorithm).
Note that the control source (GLS, DOQ, or TM6) will be selected by the infrastructure software that queries the control database and constructs the GCP library data package input to this algorithm. As such, it is not strictly an option within this algorithm, but it is an option that this processing step will select.

7.1.5.6 Procedure

This function correlates GCPs located in reference image chips to a terrain corrected Level 1G image. The GCPs are located within reference image chips. Windows are extracted from the L1G image to do the image correlation. The correlated points are written to the GCP data file for subsequent use in precision correction or product evaluation.

The heritage ALIAS and Landsat 7 implementations used L1G mensuration images that were not terrain corrected. The use of terrain corrected images reduces the size of the L1G image region that must be searched for a control point match (see maturity note #4). It also requires that the measured GCP locations be associated with elevations interpolated from the digital elevation model (DEM) used to perform the terrain correction. This is described in the LOS Model Correction ADD.

Ground Control Point Correlation Algorithm Details
GCPcorrelate performs correlations on ground control points (GCP) with a Level 1G image.

Ground Control Points
Ground control points (GCPs) and reference imagery are generated from USGS Digital Orthophoto Quadrangles (DOQs). DOQs are designed to meet national map accuracy standards at 1:24,000 scale, which corresponds to a root mean squared error (RMSE) of approximately 6 meters. A mosaic of DOQs is created by subsampling the 1 meter DOQ imagery to match the PAN band at a 15 meter resolution. Multiple DOQs are combined so that the mosaic covers a Landsat World-wide Reference System (WRS) scene extent. The ground control chip library is generated by extracting 64x64 windows from the DOQ mosaic. Since the DOQ data are only available for the United States, these GCPs cover only U.S. test sites.

Ground control chips are chosen by stepping through the DOQ imagery at evenly spaced line and sample locations. Elevation for the chips are found from DEM and stored in the GCP library. If the DOQs that comprise the mosaic have large radiometric differences, histogram equalization operations may be performed. These histogram operations include histogram matching to a reference data set or histogram balancing within the mosaic.

Ground control points have also been extracted from the Global Land Survey 2000 (GLS2000) data set. These GCPs serve as the geospatial reference for standard Landsat product generation and will be used for LDCM standard product generation. The GLS2000 GCPs provide a globally distributed, internally consistent control set. Though the GLS2000 did not include Antarctica, the Landsat Image Mosaic of Antarctica (LIMA) did, so LIMA-derived control will be used in the Antarctic regions. These GCP chips will be in the polar stereographic (PS) projection used by LIMA. Since a single projection was used for the entire continent, there are no zone issues associated with Antarctic data. The terrain corrected L1G mensuration images created for Antarctica will thus use the single LIMA polar stereographic projection. The lack of image features and prevalence of cloud cover are likely to be a more serious problem in Antarctica than elsewhere, leading to frequent GCP correlation failures in this area.

The global set of GLS chips were extracted from band 5 (SWIR1) at 30m resolution. A limited set of thermal GCPs will also be extracted from the GLS2000 ETM+ band 6 images using a selected subset of GLS scenes. These thermal GCPs will be used in the event that TIRS geometric characterization must be performed without reference to OLI data.

GCP Mensuration

Throughout the LDCM data processing and characterization algorithms, normalized cross correlation is used to measure spatial differences between two image sources. These image sources could be OLI and DOQ, OLI and Landsat, TIRS and Landsat, or OLI and OLI. Image windows are extracted and correlation is performed over the windowed area. The correlation process will only measure linear distortions over the windowed areas. By choosing windows that are well distributed throughout the imagery, nonlinear differences between the image sources can be found. Normalized cross correlation will produce a discrete correlation surface (i.e., one correlation value per integer pixel offset location). A sub pixel location associated with the offset is found by fitting a polynomial around a 3x3 area centered on the correlation peak. The polynomial coefficients can be used to solve for the peak or sub pixel location. The normalized cross correlation process helps to reduce any correlation artifacts that may arise from radiometric differences between the two image sources.

If the two image windows of size NxM are defined by f and g, the mensuration steps are:

1) Perform normalized grey scale correlation

$$R(x,y) = \frac{\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2}\left[\left(f(j,i)-\bar{f}\right)\left(g(x+j,y+i)-\bar{g}\right)\right]}{\left[\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2}\left(f(j,i)-\bar{f}\right)^2 \sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2}\left(g(x+j,y+i)-\bar{g}\right)^2\right]^{1/2}}$$

where:

$$\bar{f} = \frac{1}{(M+1)(N+1)}\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2} f(j,i)$$

$$\bar{g} = \frac{1}{(M+1)(N+1)}\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2} g(x+j,y+i)$$

R(x,y) is the grey scale discrete correlation surface.

2) Find the peak of the discrete correlation surface by searching for the integer offset with the largest correlation coefficient.

3) Fit a 2nd order polynomial to a 3x3 area centered on the correlation peak. The polynomial has the form:

$$P(x,y) = a_0 + a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2$$

A least squares fit is performed on the points to solve for the polynomial coefficients.

3a) Set up matrices

$$\underset{9x1}{[Y]} = \underset{9x6}{[X]}\,\underset{6x1}{[a]}$$

where:

X = correlation locations centered around peak
Y = correlation values corresponding to X locations
a = polynomial coefficients

Note that these matrices take the form:

$$\underset{9x1}{\begin{bmatrix} R(-1,-1) \\ R(-1,0) \\ \vdots \\ R(1,0) \\ R(1,1) \end{bmatrix}} = \underset{9x6}{\begin{bmatrix} 1 & -1 & -1 & (-1)*(-1) & (-1)^2 & (-1)^2 \\ 1 & 0 & -1 & (-1)*(0) & (0)^2 & (-1)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 1 & (1)*(0) & (0)^2 & (0)^2 \\ 1 & 1 & 1 & (1)*(1) & (1)^2 & (1)^2 \end{bmatrix}} \underset{6x1}{[a]}$$

3b) Solve for polynomial coefficients:

$$[a] = \left([X]^T [X] \right)^{-1} [X]^T [Y]$$

4) Find partial derivatives of polynomial equation in terms of x and y:

$$\frac{\partial}{\partial x} P(x,y) = a_1 + a_3 y + 2a_4 x$$
$$\frac{\partial}{\partial y} P(x,y) = a_2 + a_3 x + 2a_5 y$$

5) Set partial equations equal to zero and solve for x and y:

$$\text{x offset} = \frac{2a_1 a_5 - a_2 a_3}{a_3^2 - 4a_4 a_5}$$
$$\text{y offset} = \frac{2a_2 a_4 - a_1 a_3}{a_3^2 - 4a_4 a_5}$$

where:
x offset = sub-pixel offset in x direction
y offset = sub-pixel offset in y direction

6) Combine the sub pixel offset calculated in step 5 to the peak location from step 2 to get the total offset.

The GCP positional information and the measured sub pixel offset is recorded for each GCP along with a flag indicating whether the final correlation value passed simple correlation strength and maximum offset thresholds. No statistics-based (e.g., t-distribution) outlier detection is performed by this algorithm.

Processing Steps

The basic GCP Correlation processing flow consists of the following steps:

1. Read the GCPs and L1G image.
2. For each GCP:

2.1. Compute the predicted location of the GCP in the L1G image using the GCP map projection coordinates and any specified predicted offset.
2.2. Extract an image window from the L1G image at the predicted location.
  2.2.1. Make sure the image window contains sufficient non-fill image data.
  2.2.2. Make sure the L1G image and GCP image chip are in the same map projection (UTM zone). Reproject (see below) the GCP chip if necessary.
2.3. Correlate the GCP image chip with the L1G window to find the optimum match point.
2.4. Test the measured correlation and offset against predefined thresholds.
2.5. Write out the GCP mensuration results.

The reprojection of the GCPs in the prototype code is done as a separate step through the process called gcpretrieve. This process is a precursor to the actual correlation process. It is also worth noting that the resampling methodology is slightly different between the Landsat heritage code and the prototype. The Landsat methodology used a table of weights that were applied to each chip in order to perform the reprojection.

The GCP correlation procedure was implemented in the heritage ALIAS prototype. Though the correlation process is conceptually simple, it is computationally intensive so the ALIAS implementation was designed to be efficient. This included taking advantage of parallel processing. These processing efficiency measures make the heritage implementation somewhat more complicated than it might otherwise be. The remainder of this processing discussion follows the LDCM prototype, which was based on the heritage ALIAS implementation, to illustrate how the conceptually simple flow outlined above was mapped to a computationally efficient implementation.

7.1.5.7 Prototype Code

Input to the executable is an ODL file, output is an ASCII file containing measured offsets between the input image file and GCP library. Under the prototype output/input file directory there is a directory called chips which contains the heritage type GCP data structures and files. Under the prototype output/input directory called add that contains the ODL files needed, the HDF5 input image file, a perl script needed, and the CPF.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall -march=nocona -m32

Get GCP Correlate Parameters (get_gcpcorrelate_parameters)
This function gets parameters from the ODL parameter file.

Get GCP Information (get_gcp_information)
This function reads the GCPs from the input GCP library.

Process GCP (process_gcp)
This function processes all the GCPs by extracting the GCP image chip, extracting the image window, performing the correlation for each point, and then writing the results to the GCP data file.

NOTES:
1: The correlation routines want things in sample, line order, so the fit_offset pairs returned are horizontal (sample) and then vertical (line) offsets. In contrast, the GCP data file contains fit_offset in line, sample order.

2: To calculate the correlated location, know 2 things:

a. The correlate routines return the offset from the reference window (chip) to the search window (L1G), which is also the offset from the nominal reference point to the actual point in the search window.

b. The integer location of the predicted location roughly corresponds to the integer location of the reference location. We need to report the predicted search line, sample of the reference point and the offset from the predicted point to the correlated point. So to get the correlated location, we start with the integer location of the predicted point because this corresponds to the integer part of the ref point (this is why gcp[num_used_gcp].fit_offset subtracts the fractional part of the predicted location). Then we add the fractional part of the reference coordinate because this is really the point we are going after. Then we add the correlation fit_offset because this tells how the reference point relates to the location in the search window.

3: The calculation for fit_offset only works correctly because we are assuming the reference and search points are at the center of the window (plus some fractional distance) and the difference in window size is accounted for by nom_off; if the reference point was not at the middle of the chip, this would have to be adjusted.

Initialize Parallel Correlator (xxx_init_parallel_correlator)
This function initializes an instance of the parallel correlator. All the multiprocessing resources are created and the memory for the chip buffers and queue structures is allocated.

Get Correlation Chip Buffers (xxx_get_corr_chip_buffers)
This function returns buffers for the search and reference chips that will be submitted to the parallel correlator. Getting buffers from this routine and not submitting them to the parallel correlator will quickly exhaust all the buffers available. The buffers will be freed when the parallel correlator is closed. When compiled in single threaded mode, the same set of buffers are used for every pair of chips.

Close Parallel Correlator (xxx_close_parallel_correlator)
This function is the routine that needs to be called after all the chips have been submitted to the correlator. This routine will wait until all the threads have completed, then destroy this instance of the parallel correlator. The results of the correlation are not valid until this routine returns.

Get Search Line/Sample (get_search_line_samp)
This function finds the line and sample that corresponds to the given projection y and x. Since the L1G image is positioned (map projection) north up, finding the line (sample) is done by subtracting the upper left projection y(x) value from the GCP projection y(x) value and dividing the result by the line(sample) pixel size.

NOTES:
The line and sample numbers are 0-relative.
This will not work for a path-oriented image.

Extract Window (ias_misc_extract_window)
This function extracts an image window around a specific GCP. From the input image, a window of the specified size will be extracted around the GCP line and sample. If the window is of odd

size, the extra line and/or sample will be at the beginning of the imagery. The data in the window representing portions outside the imagery will be filled with zeros.

There are 2 steps to the extraction:
(1) data type conversion of whatever the L1G image is to float
(2) setting the calculated window correctly into the buffer (even if the calculated window falls partially outside the image)

Check Fill (oli_check_fill)
This function checks the input window for fill data over the specified percentage. This routine is useful to determine if there is too much fill data in a window. If too much fill data exists, then the window might not be good for correlating. Fill data nominally has a value of 0.0.

Extract Chip (xxx_extract_chip)
This function reads the specified image chip. The image chip is always assumed to be a flat binary file containing chip_size[0] x chip_size[1] BYTE pixels (see note #1).

Resample chip if necessary (build_gcp_lib)
New logic, derived from the Landsat Product Generation System (LPGS) heritage, will be required here to check the image chip map projection and, if necessary, resample the chip to match the L1G image. This is necessary when working with a global GCP repository containing GCPs extracted from multiple source scenes in multiple UTM zones. The GCPs falling inside a particular L1G image will frequently, particularly at high northern latitudes, be drawn from source images in neighboring UTM zones. Note that this is not a problem in Antarctica where a single polar stereographic projection is used. It is also worth noting that the resampling techniques between the LPGS heritage code and the LDCM prototype is not the same.

Image chip reprojection proceeds as follows:

1. Compute the image chip upper left (UL) corner coordinates from the GCP UTM coordinates, the GCP image line/sample coordinates, and the image chip pixel size:
    a. UL Corner X = GCP X - GCP sample coordinate * chip pixel size
    b. UL Corner Y = GCP Y + GCP line coordinate * chip pixel size

    Note that the GCP line/sample coordinates are relative to a zero-origin at the center of the upper left chip pixel.
2. Project the GCP latitude and longitude to the L1G map projection (UTM zone) using the projection transformation package (see OLI/TIRS LOS Projection ADD) to compute GCP X' and GCP Y' projected coordinates.
3. Compute the desired "new" chip UL corner in the L1G projection using the new GCP X' and GCP Y' coordinates, rounding off to a whole multiple of the pixel size:
    a. UL Corner X' = GCP X' - GCP sample coordinate * chip pixel size
    b. UL Corner Y' = GCP Y' + GCP line coordinate * chip pixel size
    c. UL Corner X' = round(UL Corner X'/chip pixel size)*chip pixel size
    d. UL Corner Y' = round(UL Corner Y'/chip pixel size)*chip pixel size
4. Compute the "new" GCP line/sample coordinates in the reprojected chip:
    a. GCP sample coordinate' = (GCP X' - UL Corner X')/chip pixel size
    b. GCP line coordinate' = (UL Corner Y' - GCP Y')/chip pixel size
5. For each point in the new chip:
    For line = 0 to nlines-1
        Compute: Y' = UL Corner Y' - line*chip pixel size

For samp = 0 to nsamps-1

1. Compute: X' = UL Corner X' + samp*chip pixel size
2. Convert (X',Y') to old chip projection (X,Y) using the projection transformation package.
3. Compute: oline = (UL Corner Y - Y)/chip pixel size
4. Compute: osamp = (X - UL Corner X)/chip pixel size
5. If the point (oline, osamp) falls inside the old chip boundary Then interpolate a DN value at that location using bilinear interpolation:
    a. lindex = (int)floor(oline)
    b. sindex = (int)floor(osamp)
    c. fline = oline - lindex
    d. fsamp = osamp - sindex
    e. DN(oline,osamp) =
        DN(lindex,sindex)*(1-fline)*(1-fsamp) +
        DN(lindex+1,sindex)*fline*(1-fsamp) +
        DN(lindex,sindex+1)*(1-fline)*fsamp +
        DN(lindex+1,sindex+1)*fline*fsamp

    Else DN(oline,osamp) = 0

5. Use the reprojected image chip and GCP line/sample coordinates in the GCP correlation procedure.

The build_gcp_lib, or gcpretreive process, is separated from the GCPcorrelate process so as to emulate the GCP retrieval process from the database containing the GCP image chips and their corresponding metadata. This retrieval process also contains the following C modules:

GCPretrieve -Main driver for GCP retrieval process.
get_gcp_lib - Reads GCPs according to set of criteria
get_gcp_information - Wrapper for reading GCPLib information
get_gcp_proj_parms - Reads projection information from image file metadata
get_gcpretrieve_parameters - Read input ODL parameters

This code was based on the Landsat ETM+/TM heritage code for GCP retrieval and chip reprojection.

Put GCP (io_put_gcp)
This function writes all GCP records to the specified output file. This function writes out a set of GCP data records. If the file already exists it will be overwritten.

Write GCP (io_write_gcp)
This function writes one record to the GCP data file. The file pointer is left at the end of the current record so sequential calls of xxx_write_gcp will sequentially write all the records in the file. The GCP data file is assumed to be an ASCII file containing one line of text per GCP data record. Each record contains: point_id, reference_line, reference_sample, latitude, longitude, elevation, predicted_search_line, predicted_search_sample, delta_y (line), delta_x (sample), accept/reject_flag, correlation coefficient, reference band number, search band number, search SCA number (0 for SCA-combined images), chip source (DOQ, GLS).

Submit Chip to Correlator (xxx_submit_chip_to_corr)
The xxx_parallel_corr module implements a parallel correlation object. Using the Posix threading interface, up to MAX_CORR_THREADS (or the number of processors available - whichever is less) are created to perform correlation. The main thread that creates the parallel correlator is then

responsible for "feeding" the parallel correlator chips to correlate. The xxx_submit_chip_to_corr places the chips into a queue. The correlation threads remove the chips from the queue and perform the correlation. The results of the correlation are not immediately available to the application since xxx_submit_chip_to_corr returns before the correlation is complete.

Before any of the correlation results are used, the application must call xxx_close_parallel_correlator to make sure all the chips have been correlated and to destroy the correlation threads.

Grey Correlator (xxx_grey_corr)
This function correlates a reference subimage with a search subimage using the pixel grey levels and evaluates the results.

Grey Cross Product Same-size (xxx_grey_cross_ss)
This function computes the unnormalized (raw) sum of pixel-by-pixel cross products between the reference and search images for every combination of horizontal and vertical offsets of the reference relative to the search image for windows of the same size (in one dimension at least).

Grey Normalization Same-size (xxx_grey_norm_ss)
This function converts raw cross-product sums to normalized cross-correlation coefficients, using tabulated statistics from previous step (grey_cross_ss). This function is much simpler than the one for unequal-sized windows, since all normalizing is done by the space domain same size correlator. All that has to be done here is statistics gathering to set up the peak finder.

Grey Cross Product (xxx_grey_cross)
This function computes the unnormalized (raw) sum of pixel-by-pixel cross products between the reference and search images for every combination of horizontal and vertical offsets of the reference relative to the search image. This function works for windows of unequal size.

Grey Normalization (xxx_grey_norm)
This function converts raw cross-product sums to normalized cross-correlation coefficients, while tabulating statistics needed for subsequent evaluation. This function works for unequal window sizes.

Grey Evaluation (xxx_grey_eval)
This function evaluates various measures of correlation validity and extracts a subarea of the cross correlation array centered on the peak.

Fit Registration (xxx_fitreg)
This function fits a quadratic surface to the neighborhood of the correlation peak and from it determine the best-fit registration offsets and their estimated errors.

Input and Output File Details

The details of the fields contained in the input GCP library file (Table 1) and the output measured GCP file (Table 2) are presented below.

Field	Description
Header Text	Zero or more lines of ASCII text, each line beginning with the "#" symbol to designate it as a header comment, describing GCP library contents.
Data Start Marker	"BEGIN" - static text to indicate beginning of data area
Number of GCPs	Integer number (N) of GCPs to follow (new)
GCP Record Fields:	One set per GCP
GCP Number	Sequence number of GCP in this package (1 to N)
GCP ID	Unique ID for GCP of the form: ppprrrnnnn where: ppprrr = WRS path/row GCP was taken from nnnn = chip sequence number
GCP Image Chip Line Coordinate	GCP location within image chip - line coordinate (fractional pixel).
GCP Image Chip Sample Coordinate	GCP location within image chip - sample coordinate (fractional pixel).
GCP Latitude	GCP latitude in degrees.
GCP Longitude	GCP longitude in degrees.
GCP X	GCP projected X coordinate in meters.
GCP Y	GCP projected Y coordinate in meters.
GCP Height	GCP WGS84 ellipsoid height in meters.
Image Chip GSD	Chip pixel size in meters.
Image Chip Lines	Number of lines in image chip.
Image Chip Samples	Number of samples in image chip.
GCP Source	Source of GCP, either "DOQ" or "GLS" or "TM6"
GCP Type	Control/validation point flag, either "CONTROL" or "VALIDATION"
Image Chip Projection	UTM or PS
Image Chip Zone	UTM zone number (1-60). Use 0 for PS.
Image Chip Date	yyyymmdd = year/month/day of GCP source
Image Chip	Link to chip image data (could be in file named with GCP ID)

Table 1: Input GCP Library Contents

Field	Description
GCP Record Fields:	One set per GCP
Point ID	GCP ID (see Table 1)
GCP chip line location	Line location of GCP within chip
GCP chip sample location	Sample location of GCP within chip
GCP latitude	GCP WGS84 latitude in degrees
GCP longitude	GCP WGS84 longitude in degrees
GCP height	GCP WGS84 ellipsoid height in meters
Predicted GCP image line	Predicted line location of GCP in L1G image
Predicted GCP image	Predicted sample location of GCP in L1G image

sample	
GCP image line offset	Measured line offset from predicted location
GCP image sample offset	Measured sample offset from predicted location
Correlation success flag	Flag 0 = correlation failure, 1 = success
Correlation coefficient	Measured correlation coefficient (new)
Search band number	L1G band number used
Search SCA number	L1G SCA where GCP was found
Chip source	GCP source (DOQ or GLS or TM6)

Table 2: Output GCP Mensuration File Contents

7.1.5.8 Maturity

Though much of the ALI model correction algorithm will be reusable there are several areas where changes are expected:

5. The heritage process takes WRS path/row as input and accesses a static GCP Library file set indexed by path/row. For the LDCM implementation it is assumed that GCP storage and retrieval is managed externally and that this process will be provided with a set of GCPs applicable to the image area. The prototype uses a pre-processing step involving a perl script and a C executable called gcpretrieve to mimic the database retrieval and chip reprojection steps. These steps are discussed in the prototype and verification sections.
6. The computed correlation coefficient is added as an output to make it available for subsequent outlier filtering, if necessary.
7. At high latitudes scenes will frequently straddle multiple UTM zones. The control point chips falling in a given scene may thus be in more than one projection leading to difficulties in correlating the chips (due to the rotation between the chip projection and the mensuration image projection. This problem has been solved in the Landsat processing system (LPGS) by including logic in gcpcorrelate that resamples the GCP chips, if necessary, to match the mensuration image projection. This logic is not part of the ALIAS heritage code but will be needed to support global LDCM product generation using the GLS ground control. The reprojection of the chips has been addressed and prototyped. This is no longer and issue with the prototype code.
8. The baseline plan for LDCM GCP correlation is to use a terrain corrected, rather than the heritage systematically corrected, image for GCP mensuration. This is a departure from the ALIAS (and Landsat) heritage, motivated by the implications of processing off-nadir images.
    a. The predicted GCP locations used to establish the search area in the mensuration image are computed without reference to terrain displacement effects. For nadir images the terrain-induced cross-track offsets introduced between these “flat Earth” predictions and the actual GCP locations are small enough to fit inside the normal GCP search window: a maximum elevation of 8000m corresponds to ~1200m of parallax at the edge of the swath which is a displacement of ~40 pixels (at the 30m GLS control resolution). A 64x64 GCP chip and a 128x128 search area can accommodate offsets up to +/-31 pixels whereas a 256x256 search area can accommodate offsets up to +/-95 pixels.
    b. For off-nadir images the terrain sensitivity is roughly tripled so offsets up to 120 pixels would have to be accounted for. Since increasing the search area substantially increases processing time and increases the likelihood of a false GCP match, and given the fact that LDCM pointing should be sufficiently accurate to generate predicted GCP locations that are within a pixel or two most of the time, it would be better to account for terrain offsets when predicting GCP locations to minimize the search area. One way to

do this would be to calculate the magnitude of this effect and include it in the computation of the predicted GCP locations. The problem with this is that the LOS model correction algorithm uses the measured offset between predicted and measured GCP locations to derive model corrections, so any adjustment to the predicted location based on a computed terrain offset, will have to be accounted for in the precision correction algorithm. Thus, complicated offset prediction and offset removal logic would need to be added to both the GCP correlation and LOS model correction algorithms.

c. A better way to accomplish the same objective is to perform the mensuration on a terrain corrected image, where the terrain offsets have been accounted for explicitly in the image generation process. This approach is preferable to attempting to correct for terrain point-by-point in the mensuration of a systematically corrected image, but it does have some drawbacks:
    i. Using a terrain corrected mensuration image will require using the DEM as another input to the LOS model correction algorithm in order to compensate for any difference between the GCP elevation and the corresponding DEM elevation at the point where the GCP match was found in the mensuration image. This adds complexity to an already complex algorithm.
    ii. Misregistration between the systematic image and the DEM can cause the terrain correction process to inject high frequency image distortion. This would probably not be a huge problem given the expected accuracy of LDCM pointing and ephemeris knowledge. In areas of significant relief, even a slightly misregistered DEM may provide a mensuration image that is more similar to the GCP chips, which are themselves terrain corrected, than a systematic image would be.

7.1.5.9 Notes

Some additional background assumptions and notes include:

1. The heritage GLS, TM6, DOQ image chips are stored as 8-bit (BYTE) arrays whereas the LDCM imagery will be 16-bit (or float). The correlation is performed on floating point data so both the image and the chips are converted to float on input. Thus the image and chip data types need not match.
2. The correlation result fit method defines the algorithm used to estimate the correlation peak location to sub-pixel accuracy. Only the quadratic surface fitting method described in this ADD is supported in the baseline algorithm.
3. Though the normal baseline for measuring control points is to use an SCA-separated terrain corrected image, this algorithm should also function with a combined-SCA image so that it can be used to measure test point GCPs in L1T product images to support the geometric accuracy characterization algorithm.
4. The GCPs in the GCP repository (part of the Infrastructure Element) should be flagged as either “control” points, to be used for LOS model correction, or “validation” points, to be used for geometric accuracy characterization. The utility that extracts control points from this repository should be able to extract either control set. The “control” set would contain the majority of the points. The “validation” flag would only be used in areas where more than some minimum threshold number of GCPs are available. These flags would be set by the Cal/Val team at the time the GCP repository was loaded and could be adjusted, if necessary, thereafter. Criteria for selecting validation points would be based upon considerations such as:
    a. The total number of available GCPs in the scene must exceed some minimum (e.g., 100).
    b. Points that fall on the boundary (or, more precisely, the convex hull) of the GCP set would not be validation point candidates.

c. Points that are within some maximum distance (e.g., 25 km) of another GCP would be validation point candidates.

The goal would be to develop an automated validation point identification algorithm that would operate somewhat like an outlier rejection algorithm: identify the best validation point candidate based upon a set of criteria, remove it from the control point list, and iterate until no additional validation points are identified.

5. Scenes with poor geolocation accuracy can lead to the actual GCP L1G image locations being sufficiently far from their predicted locations so as to make it impractical to expand the GCP search window to the extent necessary to find the GCPs. An optional parameter to specify an a priori predicted offset provides a more reliable way to find and correctly correlate the GCPs in this situation. This can occur early in the mission, before the first on-orbit sensor alignment calibration, or during an anomaly investigation.

7.2 OLI Geometry Algorithms

7.2.1 OLI Line-of-Sight Model Creation Algorithm

7.2.1.1 Background/Introduction

The line-of-sight (LOS) model creation algorithm gathers the ancillary data and calibration parameters required to support geometric processing of the input image data set, validates the image time codes, extracts the corresponding ephemeris and attitude data from the ancillary data stream, performs the necessary coordinate transformations, and stores the results in a geometric model structure for subsequent use by other geometric algorithms. The OLI LOS model creation algorithm is derived from the ALI model creation algorithm used in ALIAS. Its implementation is very similar to the alinit application and the geometric sensor (axx) and spacecraft (exx) model libraries used in ALIAS, though much of the ephemeris and attitude preprocessing logic present in alinit has been moved to the, now separate, ancillary data preprocessing algorithm to better isolate the bulk of the geometric processing logic from the details of the incoming ancillary data stream. New attitude data processing logic has also been added to separate the high- and low-frequency attitude effects to allow the image resampling process to better correct for jitter at frequencies above the original 10 Hz algorithm design limit without requiring an unreasonably dense resampling grid.

7.2.1.2 Dependencies

The LOS Model Creation algorithm assumes that the Ancillary Data Preprocessing algorithm has been executed to accomplish the following:

Validated ephemeris data for the full imaging interval have been generated
Validated attitude data for the full imaging interval have been generated
The ancillary data have been scaled to engineering units

Whether or not “definitive” processing has been performed, the Ancillary Data Preprocessing algorithm will generate preprocessed smoothed and cleaned ephemeris and attitude data streams. The format will be the same for either validated spacecraft estimates or definitive processing.

7.2.1.3 Inputs

The LOS Model Creation algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data; including data set IDs to provide unique identifiers for data trending).

Algorithm Inputs
ODL File (implementation)
Acquisition Type (Earth, Lunar, Stellar) (optional, defaults to Earth)
CPF File Name
Ancillary Data Input File Name
L0R/L1R Directory and File Names
WRS Path/Row (stored in model and used for trending)
Trending On/Off Switch (not implemented in prototype)
L0Rp ID (for trending)
Work Order ID (for trending)
Optional Precision Model Input Parameters (see note 9)
Input Precision Model Reference Time (optional)
Input Precision Ephemeris Correction Order (optional)
Input Precision X Correction Parameters (optional)
Input Precision Y Correction Parameters (optional)
Input Precision Z Correction Parameters (optional)
Input Precision Attitude Correction Order (optional)
Input Precision Roll Correction Parameters (optional)
Input Precision Pitch Correction Parameters (optional)
Input Precision Yaw Correction Parameters (optional)
CPF Contents
WGS84 Earth ellipsoid parameters
Earth orientation parameters (UT1UTC, pole wander, leap seconds) (see note 1)
Earth rotation velocity
Speed of light
ACS to OLI rotation matrix
Spacecraft center of mass (CM) to OLI offset in ACS reference frame (meters)
High frequency attitude data cutoff frequency (Hz)
Focal plane model parameters (Legendre coefficients)
Detector delay table (now including whole pixel even/odd and deselect offsets)
Nominal L0R fill (per band)
Nominal OLI frame time nominal_frame_time (4.236 msec)
Nominal OLI MS and pan integration times (msec)
OLI MS and pan detector settling times (msec)
Preprocessed Ancillary Data Contents
Attitude Data
Attitude data UTC epoch: Year, Day of Year, Seconds of Day
Time from epoch (one per sample, nominally 50 Hz)
ECI quaternion (vector: q1, q2, q3, scalar: q4) (one per sample)
ECEF quaternion (one per sample)
Body rate estimate (roll, pitch, yaw rate) (one per sample)
Roll, pitch, yaw estimate (one per sample)
Ephemeris Data
Ephemeris data UTC epoch: Year, Day of Year, Seconds of Day
Time from epoch (one per sample, nominally 1 Hz)
ECI position estimate (X, Y, Z) (one set per sample)
ECI velocity estimate (Vx, Vy, Vz) (one set per sample)
ECEF position estimate (X, Y, Z) (one set per sample)
ECEF velocity estimate (Vx, Vy, Vz) (one set per sample)
L0R/L1R Data Contents
Image Time Codes (one per line)
Integration Time (one value for MS bands and one value for pan band)
Detector Alignment Fill Table (see note 2)

7.2.1.4 Outputs

OLI LOS Model (additional detail is provided in Table 1 below)
WGS84 Earth ellipsoid parameters
Earth Orientation Parameters (for current day) from CPF
Earth rotation velocity
Speed of light
OLI to ACS reference alignment matrix/quaternion
Spacecraft center of mass to OLI offset in ACS reference frame
Focal plane model parameters (Legendre coefs)
Detector delay table (now including whole pixel even/odd and deselect offsets)
Nominal detector alignment fill table (from CPF)
L0R detector alignment fill table (from L0R)
ECI J2000 spacecraft ephemeris model (original and corrected)
ECEF spacecraft ephemeris model (original and corrected)
Spacecraft attitude model (time, roll, pitch, yaw) (orig and corr) (see note 4)
High frequency attitude perturbations (roll, pitch, yaw) per image line (jitter table)
Image time codes (see note 5) (in seconds)
Integration Time (MS and pan) (in seconds)
Sample Time (MS and pan) (in seconds)
Settling Time (MS and pan) (in seconds)
WRS Path/Row
Model Trending Data
WRS Path/Row
L0Rp ID
Work Order ID
Image start UTC time (year, day of year, seconds of day)
Computed image frame time (in seconds)
Number of image lines
Number of out of limit image time codes

7.2.1.5 Options

Trending On/Off Switch
Optional precision model input parameters can be used to force model corrections.

7.2.1.6 Prototype Code

Input to the executable is an ODL file; output is a HDF4 formatted OLI model file.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The following text is a brief description of the main set of modules used within the prototype with each module listed along with a very short description. It should be noted that not all library modules are referenced in the explanations below. The modules within the main create directory of the prototype are discussed and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

model_create – Main procedure that retrieves the input parameters and invokes the model generation and model output logic.

getpar – Retreives the user-provided ODL parameters.

oli_zero_model – Library routine that initializes the model structure.

get_path_row_l0ra - Designed to retrieve the WRS path and row numbers from the L0R data. In the baseline algorithm these are ODL input parameters but they should ultimately be extracted from the Level 0R data directly. This unit is a placeholder for the time being.

oli_run_model – Library routine that loads the CPF, L0R, and preprocessed ancillary data into the model structure.

- oli_get_cpf – Library routine that reads the CPF.
- oli_get_model_sensor_params – Library routine that loads the sensor section of the model structure using data from the CPF and the L0R frame header.
- oli_get_model_image_params – Library routine that loads the image section of the model structure using data from the CPF, the L0R line headers, and the L0R detector offset fields. This unit also validates the image line time codes.
- oli_get_model_earth_params – Library routine that loads the Earth model parameters from the CPF.
- oli_get_ancillary_pre – Library routine that loads the attitude and ephemeris model sections using data from the preprocessed ancillary data file.
- oli_build_jitter_table – Library routine that splits the attitude data from the ancillary data into low- and high-frequency streams, interpolates the high frequency data to match the OLI panchromatic band line times, stores this per image line high frequency attitude data in the jitter table structure, and replaces the original combined attitude data stream with the low-frequency stream.
    - remez – Library routine that uses the Remez exchange algorithm to synthesize the weights (taps) of a low pass finite impulse response digital filter based on input filter size and cutoff frequency parameters. GNU Public License code written by Jake Janovetz, formerly of UIUC, which is available online at his site: http://www.janovetz.com/jake/ and more specifically: http://www.janovetz.com/jake/remez/remez-19980711.zip

l8_correct_attitude – Library routine that applies the user-input precision model attitude corrections (if any).

l8_convert_ephem – Library routine that applies the user-input precision model ephemeris corrections (if any).

oli_put_model – Library routine that writes the OLI model structure to the output HDF model file.

7.2.1.7 Procedure

The LOS model is stored as a structure and is created from information contained in the Level 0R or Level 1R image data, the CPF, and the Ancillary data. The model is subsequently used along with the CPF to create a resampling grid. Data present in the model structure includes satellite position, velocity, and attitude, line-of-sight (LOS) angles, timing references, precision correction information (if any), and the software version. The LOS model is also used in several characterization and calibration routines for mapping input line/sample locations to geographic latitude/longitude.

The LOS model may be thought of in two parts, an instrument model that provides a line-of-sight vector for each OLI detector (and, hence, each image line/sample), and a spacecraft model that provides spacecraft ephemeris (position and velocity) and attitude as a function of time. These models are linked by the image time stamps that allow each Level 0R or Level 1R image sample to be associated with a time of observation.

7.2.1.7.1 Instrument Model

The model treats every band of every SCA independently. This is done by defining a set of 2nd order Legendre polynomials for each band of each SCA. Since the odd and even detectors are staggered for each band (Figure 1) as well as there being multiple pixel selects, the set of Legendre polynomials represent a theoretical “nominal” set of detectors that are best-fit to the even detectors for the first pixel select. This approach treats the odd detectors and second and third pixel select detectors as though they were aligned with the even detectors for the first pixel select for purposes of sensor LOS generation. This approach also explicitly models the slight offsets caused by the actual odd detector offset, any offsets caused by detector deselect, and the sub-pixel deviations of each detector from its nominal location, for correction during image resampling. This leads to four detector types: nominal, actual, maximum, and exact. A nominal detector is calculated from the Legendre polynomials. An actual detector corrects the nominal detector location for the whole pixel odd/even and pixel select offsets. For the ALI, these offsets were band dependent. For the OLI, since individual detectors may be deselected, they are detector dependent. The maximum detector option uses the largest possible even/odd and pixel select offset for a given band. This is used to compute detector terrain parallax sensitivity coefficients when generating the line-of-sight grid. See the LOS Projection/Grid Generation Algorithm Description Document for additional details. An exact detector has the actual correction applied but also includes the specific individual (sub-pixel) detector offsets. The Legendre polynomials and a table of detector offset values are stored in the CPF.

There is a slight angular difference between the line of sight vectors or angles associated with the odd/even and multiple pixel select detectors. If the nominal LOS, generated using the 2nd order Legendre model, is $\psi_{nominal}$, the look angles for the actual and exact detectors are:

$$\psi_{x\_actual} = \psi_{x\_nominal} + \text{round}(\text{detector\_shift\_x}) * \text{IFOV}$$
$$\psi_{y\_actual} = \psi_{y\_nominal} + \text{round}(\text{detector\_shift\_y}) * \text{IFOV}$$

$$\psi_{x\_exact} = \psi_{x\_nominal} + \text{detector\_shift\_x} * \text{IFOV}$$
$$\psi_{y\_exact} = \psi_{y\_nominal} + \text{detector\_shift\_y} * \text{IFOV}$$

The maximum detector case uses the largest possible along-track offset and the nominal across-track offset, which is zero:

$$\psi_{x\_maximum} = \psi_{x\_nominal} + \text{maximum\_shift\_x} * \text{IFOV}$$
$$\psi_{y\_maximum} = \psi_{y\_nominal}$$

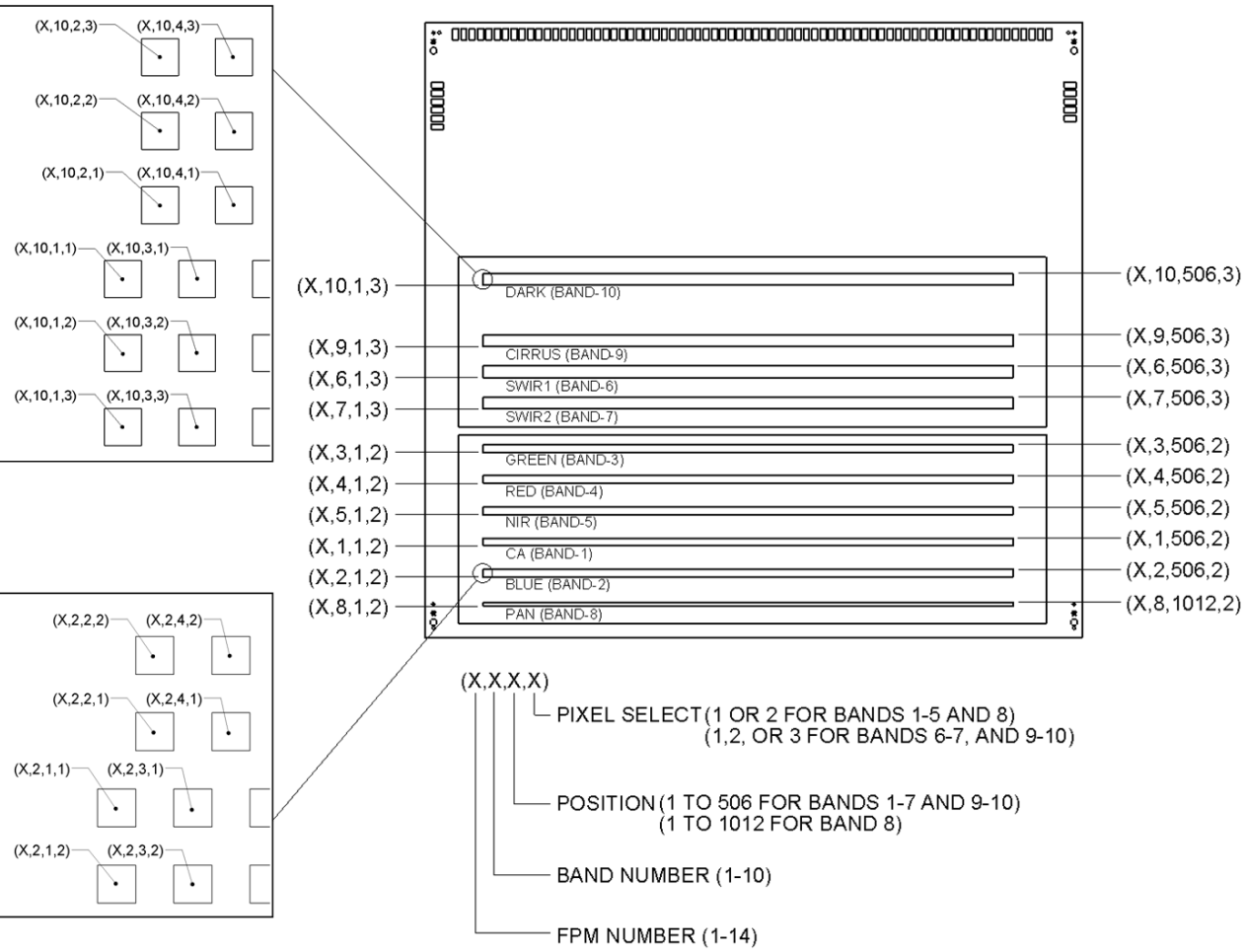


Figure 1: Detector Layout

The detector_shift_x and detector_shift_y values are the detector-specific offsets stored in the CPF detector delay tables. These offsets include both the whole-pixel even/odd and deselect offsets and the fractional-pixel detector placement effects, and must be rounded to extract the integer portion. Note that the integer portion of the detector_shift_y value is always zero since the even/odd and deselect effects are applicable only in the X direction.

The nominal LOS is used in most line-of-sight projection applications. The actual LOS is used in conjunction with the actual image time (see below) to model the errors introduced by trading time (sample delay) for space (detector offset) for purposes of correcting the nominal LOS model. The exact LOS is generally used only for data simulation and other analytical purposes rather than in the geometric correction model, as the sub-pixel portion of the detector delay is applied directly in the image resampler rather than being included in the LOS model.

7.2.1.7.2 Sample Timing

The OLI provides a time stamp with each image frame collected. These time stamps make it possible to relate the image samples (pixels) to the corresponding spacecraft ephemeris and attitude data. The OLI sample timing relationships are shown in Figure 2. Several items in this figure are worthy of

particular note. First, the time stamp associated with a data frame is recorded at the end of the detector integration time. Second, there is a small settling and sampling delay (MS SS and Pan SS in the figure) between the end of detector integration and time stamp generation. Third, the time stamps are delayed by one data frame so that date frame N contains the time stamp for data frame N-1. Fourth, the data frame associated with time stamp N contains the multispectral (MS) samples collected just prior to time stamp N as well as the panchromatic samples collected just prior to and just after time stamp N, rather than the two samples collected prior to time stamp N. This is important for relating the panchromatic sample timing to the multispectral sample timing.

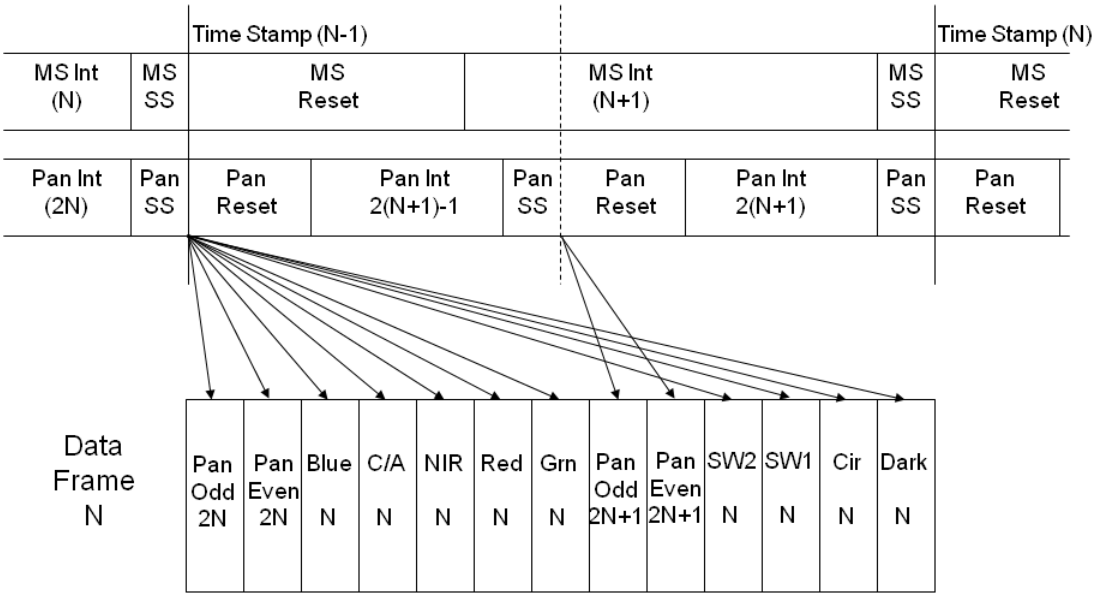


Figure 2: OLI Focal Plane Electronics Detector Sample Timing Diagram

One further complication to the problem of assigning times to image samples is the fact that the Level 0R/1R input imagery will include fill pixels inserted to achieve nominal even/odd detector alignment. This fill insertion allows the geometrically unprocessed 0R/1R imagery to be viewed as a spatially contiguous image without even/odd detector misalignments. The amount of detector alignment fill present will be indicated in the L0R/L1R image data (this is the purpose of the detector alignment fill table input noted above) so that the association of image samples with their corresponding time stamps can be adjusted accordingly. In the heritage ALIAS system fill pixels were also inserted to achieve nominal band alignment. The assumption here is that this will not be done for OLI data.

Due to the staggered odd/even and multiple pixel select detectors, a nominal and an actual time can be found in a scene. The actual time reflects the time that the current detector was actually sampled whereas the nominal time reflects the time at which the idealized detector represented by the OLI LOS model would have been sampled. There is also a “maximum” detector time option used in the computation of detector terrain parallax sensitivity coefficients during grid generation.

If the current position within the image is given as a line and sample location, the two different “types” of times for multispectral pixels are calculated by:

```
if detector_type is set to MAXIMUM
    l0r_fill_pixels = nominal_fill_pixels + round(maximum_detector_delay)
else
    l0r_fill_pixels = Fill value for current detector from L0Rp
```

```
time_index = round( MS_line ) - l0r_fill_pixels + 1
if ( time_index < 0 ) time_index = 0
if (time_index > (num_time_stamps - 1)) time_index = num_time_stamps - 1

MS_actual_time = line_time_stamp[time_index] - MS_settle_time
                 - MS_integration_time/2
                 + (MS_line - l0r_fill_pixels – (time_index-1)) * MS_sample_time

MS_nominal_time = MS_actual_time
                  + (l0r_fill_pixels – nominal_fill_pixels) * MS_sample_time
```

where:

- MS_line is the zero-referenced multispectral line number (N).
- nominal_fill_pixels is the amount of even/odd detector alignment fill to be inserted at the beginning of pixel columns that correspond to nominal detectors; that is, those detectors with a delay value of zero that are the basis for the Legendre polynomial LOS model. This value comes from the CPF.
- l0r_fill_pixels is the total amount of even/odd detector alignment fill inserted at the beginning of the pixel column associated with the current detector in the Level 0R image data. It includes both the nominal_fill_pixels and the detector-specific delay fill required to align even/odd detectors.
- num_time_stamps is the total number of time codes (data frames) in the image. It is tested to ensure that time_index, the line_time_stamp index, does not go out of bounds.
- maximum_detector_delay is a constant offset that represents the largest amount of even/odd detector offset for any detector from the LOS model detector delay table. It is rounded to the nearest integer pixel because time offsets can only occur in whole line increments. The value of this parameter is not critical as the line-of-sight offsets computed for “maximum” detectors are divided by the maximum delay to compute offset-per-unit-delay coefficients. This parameter is set in a #define statement.
- MS_settle_time is a small sample and hold time delay constant.

The MS_settle_time correction is expected to be a small (tens of microseconds) constant offset that will be captured in the CPF. The L0R/L1R data can be accessed by SCA making the association of sample number with detector index more straightforward. Note that the Level 0R data inverts the detector read-out order for the even numbered SCAs so that the samples are numbered left-to-right for all SCAs (see note 6). This convention is also followed in the CPF detector offset tables. Also note that the non-imaging detector data (video reference pixels) are stored separately from the image data in the L0R and are also not modeled in the CPF (see note 7). There are thus 494 samples per SCA in the multispectral bands and 988 samples per SCA in the panchromatic band.

For the panchromatic band the corresponding equations for a pan detector in the two pan lines (2N and 2N+1) associated with MS line N (reference Figure 2) are computed as:

```
if detector_type is set to MAXIMUM
    l0r_fill_pixels = nominal_fill_pixels + round(maximum_detector_delay)
else
    l0r_fill_pixels = Fill value for current detector from L0Rp
```

```
time_index = floor( (round( pan_line ) - l0r_fill_pixels)/2 ) + 1
if ( time_index < 0 ) time_index = 0
if (time_index > (num_time_stamps - 1)) time_index = num_time_stamps - 1

Pan_actual_time = line_time_stamp[time_index] - Pan_settle_time
                  - Pan_integration_time/2
                  + (pan_line - l0r_fill_pixels - 2*(time_index-1))*Pan_sample_time

Pan_nominal_time = Pan_actual_time
                   + (l0r_fill_pixels – nominal_fill_pixels) * Pan_sample_time
```

where:

- pan_line is the zero-referenced panchromatic line number (2N or 2N+1).
- nominal_fill_pixels is the amount of even/odd detector alignment fill to be inserted at the beginning of pixel columns that correspond to nominal detectors; that is, those detectors with a delay value of zero that are the basis for the Legendre polynomial LOS model. This value comes from the CPF.
- l0r_fill_pixels is the total amount of even/odd detector alignment fill to be inserted at the beginning of the pixel column associated with the current detector. It includes both the nominal_fill_pixels and the detector-specific delay fill required to align even/odd detectors. Note that these values will always be even for the panchromatic band.
- num_time_stamps is the total number of time codes (data frames) in the image. It is tested to ensure that time_index, the line_time_stamp index, does not go out of bounds.
- maximum_detector_delay is a constant offset that represents the largest amount of even/odd detector offset for any detector from the LOS model detector delay table. It is rounded to the nearest integer pixel because time offsets can only occur in whole line increments. The value of this parameter is not critical as the line-of-sight offsets computed for “maximum” detectors are divided by the maximum delay to compute offset-per-unit-delay coefficients. This parameter is set in a #define statement.
- Pan_settle_time is a small sample and hold time delay constant.

For the panchromatic band, the l0r_fill_pixels, nominal_fill_pixels, and maximum_detector_delay parameters are in units of panchromatic pixels.

Note that when fill is used to align even and odd detectors the spatial difference between the nominal and actual look vectors is approximately compensated by the time difference between $t_{nominal}$ and t_{actual}.

Spacecraft Model
The spacecraft ephemeris and attitude models are constructed from the input preprocessed ancillary data by extracting the ancillary data that span the current image. Both ECI and ECEF versions of the ephemeris data are retained in the model structure to avoid the need to repeatedly invoke the ECI/ECEF coordinate system conversion. The ALIAS heritage roll-pitch-yaw representation of the attitude model is retained in the model structure though a quaternion representation may be used in a future algorithm revision (see note 4).

Prepare LOS Model Sub-Algorithm

This function gathers the information from the preprocessed ancillary data and the Level 0R/1R image data needed to process model data and run the LOS model. Though it has the same overall purpose and function as the heritage axx_run_alimodel unit, differences in the details of how image timing and spacecraft telemetry information are provided for LDCM, as compared to EO-1, lead to extensive changes.

The main steps are:

1. Load the image time codes and convert to seconds since spacecraft epoch.
2. Determine the image time window.
3. Validate/smooth the image time codes.
4. Extract the multispectral and panchromatic integration times from the Level 0R/1R image frame header data.
5. Extract the associated ephemeris and attitude data from the preprocessed ancillary data stream.
6. Preprocess the input attitude data into a low-frequency stream, used for basic geometric modeling, and a high-frequency stream, used as a fine correction in the image resampler. This preprocessing was added to improve the ability of the geometric correction system to compensate for jitter disturbance frequencies above 10 Hz.

The input preprocessed ancillary data are stored in an HDF file. The attitude and ephemeris ancillary data streams each have an epoch time identifying the UTC date/time reference. Within these data streams, each attitude or ephemeris observation in the HDF file has a corresponding time offset relative to the epoch. This incoming ancillary data stream spans the entire imaging interval containing the image data represented in the Level 0R/1R input data. In creating the model we identify and extract the ancillary data sequence required to process the current image data.

The input Level 0R/1R image data are also packaged in HDF files that include the image samples for each band and SCA and the time codes assigned to each image line by the OLI instrument. These spacecraft time codes are provided by the OLI in CCSDS T-Field format which includes days since epoch (16-bit integer), milliseconds of day (32-bit integer) and microseconds of millisecond (16-bit integer) fields:

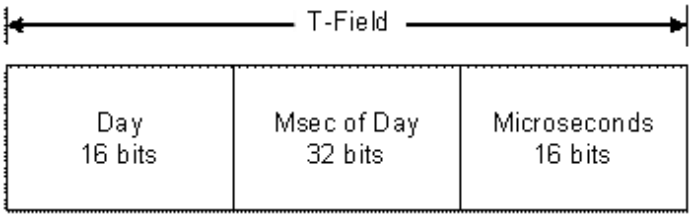


Figure 3: OLI Time Code Format

Level 0 processing will combine these raw time code fields to compute time since the spacecraft epoch in the form: days since spacecraft epoch and seconds of day. Since they are derived from the spacecraft clock, the image time codes will be based on the same epoch used by the ancillary data (e.g., TAI seconds from J2000). Even though the initial time code conversion will occur in Level 0 processing, for completeness the processing described below begins with the raw time code fields shown in figure 3.

Process Image Time Codes

The image time codes are loaded from the input HDF Level 0R/1R data set. Even/odd detector alignment fill may be inserted into the Level 0R/1R imagery as described above, so the image lines

each contain samples collected at times that may be offset from the time specified by the corresponding time code. The relationship between these time codes, the OLI integration times, and the multispectral and panchromatic pixel center times has already been described above. The L0Rp data will contain one time code per multispectral image line, excluding fill, or a nominal 6701 time codes per scene. The image files themselves may be up to 10 lines longer to accommodate the even/odd detector alignment fill.

A defect in the OLI timing logic can lead to erroneous time codes being generated when the microseconds or milliseconds fields fail to roll over properly. In the first case, the microseconds field can reach 1000 and increment the millisecond field without rolling over to zero. In the second case, the milliseconds field can reach 86400000 and increment the day field without rolling over to zero. Though these errors should be detected and corrected during Level 0 processing the following time code validation logic will detect and correct this effect as well as other suspicious time codes.

1. Convert the time codes to seconds from spacecraft epoch:
   Line_time = TC_Day*86400 + TC_MSec/1000 + TC_Micro/1e6
   Note that an IEEE 754 double precision (64-bit) number with a 52-bit fraction should provide sufficient precision to represent time differences from 01JAN2000 to 01JAN2050 with microsecond accuracy (1.6e15 microseconds < 2^51).
2. Validate the image time codes as follows:
   a. Loop through the time codes from 1 to N-1, where N is the number of image data frames/time codes, and test the difference between the current and previous time codes against the nominal frame time from the CPF using the #define tolerance DTIME_TOL. The first of two consecutive time codes that are within the tolerance is the first valid time code.
   b. Initialize the OLI clock model by setting the least squares variables to zero: $A_{00} = A_{01} = A_{11} = L_0 = L_1 = 0$
      i. Since the normal equation matrix, A, is symmetric, $A_{10} = A_{01}$ so it is not computed separately.
      ii. Add the first valid time code observation by adding 1 to A_{00}. This is all that is required since, by definition, the index difference and time difference (see below) are zero at the first valid point.
   c. For each subsequent time code:
      i. Compare the time difference from the previous time code to the nominal value using the DTIME_TOL threshold.
      ii. If a time code fails this check, see if the special conditions of the known OLI time code defect apply:
         1. If the time code difference deviates from the nominal value by more than 0.5 milliseconds:
            a. If the time code microseconds field = 1000, subtract 1000
            b. If the time code milliseconds field = 86400000 and the microseconds field = 0, set the milliseconds field to zero.
            c. Recalculate the time code difference
      iii. Compare the time code difference to a larger outlier tolerance (OUTLIER_TOL) chosen to bound the possible drift in the OLI clock relative to the spacecraft clock (currently set to 50 microsec).
      iv. If the time code difference is within the outlier range, add the current time to a least squares linear OLI clock model:
         1. Δnum = current index – first valid index

2. Δtime = current time – first valid time
3. Accumulate:
    a. Valid point count: A_{00} += 1
    b. Index difference: A_{01} += Δnum
    c. Squared index diff: A_{11} += Δnum*Δnum
    d. Time difference: L_0 += Δtime
    e. Time diff*index diff: L_1 += Δnum*Δtime

d. Once all time codes have been analyzed, solve for the linear OLI clock model parameters:
    i. determinant = $A_{00}*A_{11} – A_{01}*A_{01}$
    ii. If abs(determinant) <= 0.0 return an error
    iii. Offset = first valid time + $(A_{11}*L_0 – A_{01}*L_1)$ / determinant
    iv. Rate = $(A_{00}*L_1 – A_{01}*L_0)$ / determinant
e. Use the correction model to replace bad time codes:
    i. For each time code:
        1. Calculate the corresponding model time as:
        Mtime = Offset + (current index – first valid index) * Rate
        2. Calculate the actual time – model time difference.
        Diff = abs(time code – Mtime)
        3. Test the difference against DTIME_TOL
        4. If the difference exceeds DTIME_TOL, replace the current time code with the model value, Mtime
f. If no valid time codes were found, return an error.
g. Calculate the average observed frame time, delta_time, by subtracting the first valid/corrected time code from the last valid/corrected time code and dividing by the number of time code minus one.
h. Store delta_time (MS frame time) and delta_time/2 (pan frame time) in the model.

3. Set the image start time: image_start = line_time[0]
4. Subtract the image start time from the line time codes so that the times are seconds from image start.
5. Store the image start UTC epoch (image_year, image_day, image_seconds) and the image line offset times in the model structure.
6. Report/trend the results of the time code processing including:
    a. WRS Path/Row (input parameters)
    b. Image UTC epoch (year, day, seconds of day)
    c. L0R ID (input parameter)
    d. Work order ID (input parameter)
    e. Computed frame time (delta_time)
    f. Number of replaced time codes (bad_image_time_count)
7. Check the pan and MS detector integration times in the L0Rp frame header and if they are valid (> 0), convert them to units of seconds and load them in the model. Otherwise use the nominal values from the CPF converted to units of seconds.
8. Load the detector settling times from the CPF into the model after converting them to units of seconds.

Extract Ancillary Ephemeris and Attitude Data

The subset of ancillary ephemeris and attitude data needed to span the image data are extracted from the Level 0R data by the ancillary data preprocessing algorithm. The logic to do the required subsetting is reiterated below for reference, since that phase 3 algorithm has not yet been released.

These data are read from the input preprocessed ancillary data stream and stored in the model structure during model creation.

The ephemeris data extraction/subsetting procedure is as follows:

1. Compute the time offset from the ephemeris epoch time to the desired ephemeris start time for this image.
   ephem_start = image_seconds – ancillary_overlap – ephem_seconds
   Noting that image_seconds and ephem_seconds are the seconds of day fields from the image and ephemeris epoch times, respectively, and ancillary_overlap is the desired extra ancillary data before and after the image window (set in a #define statement).
2. Loop through the ephemeris sample times to find the last entry that does not exceed ephem_start. This is the ephemeris start index (eph_start_index).
3. Compute the time offset from the ephemeris epoch time to the desired ephemeris stop time for this image.
   ephem_stop = image_seconds + line_time[N-1] + ancillary_overlap – ephem_seconds
4. Loop through the ephemeris sample times to find the first entry that exceeds ephem_stop. This is the ephemeris stop index (eph_stop_index).
5. Compute a new ephemeris UTC epoch for this image:
   imgeph_year = ephem_year
   imgeph_day = ephem_day
   imgeph_seconds = ephem_seconds + ephem_samp_time[eph_start_index]
6. Load the ECI and ECEF ephemeris samples from eph_start_index to eph_stop_index (inclusive) into the preprocessed ancillary data output, adjusting the sample times so that they are offset from the UTC epoch computed in step 5.

The attitude data extraction/subsetting procedure is as follows:

1. Compute the time offset from the attitude epoch time to the desired attitude start time for this image.
   att_start = image_seconds – ancillary_overlap – att_seconds
   Noting that image_seconds and att_seconds are the seconds of day fields from the image and attitude epoch times, respectively.
2. Loop through the attitude sample times to find the last entry that does not exceed att_start. This is the attitude start index (att_start_index).
3. Compute the time offset from the attitude epoch time to the desired attitude stop time for this image.
   att_stop = image_seconds + line_time[N-1] + ancillary_overlap – att_seconds
4. Loop through the attitude sample times to find the first entry that exceeds att_stop. This is the attitude stop index (att_stop_index).
5. Compute a new attitude UTC epoch for this image:
   imgatt_year = att_year
   imgatt_day = att_day
   imgatt_seconds = att_seconds + att_samp_time[att_start_index]
6. For Earth-view acquisitions, load the roll-pitch-yaw samples from att_start_index to att_stop_index (inclusive) into the preprocessed ancillary data output, adjusting the sample times so that they are offset from the UTC epoch computed in step 5.
7. For lunar/stellar acquisitions, convert the ECI quaternion samples from att_start_index to att_stop_index (inclusive) to ECI roll-pitch-yaw values, as described below, and store the computed roll-pitch-yaw values in the output, adjusting the sample times so that they are offset from the UTC epoch computed in step 5.

Converting ECI Quaternions to Roll-Pitch-Yaw

For lunar and stellar acquisitions, the ECI attitude representation is stored in the model structure. In the baseline model, this is done by converting the ECI quaternions to roll-pitch-yaw values relative to the ECI axes. This is one of the motivations for considering a transition to using a quaternion attitude representation in the model in the future.

The ECI quaternions are converted to roll-pitch-yaw values as follows:

1. Compute the rotation matrix corresponding to the ECI quaternion values:

$\mathbf{M}_{ACS2ECI} =$

$$\begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix}$$

2. Compute the corresponding ACS to ECI roll-pitch-yaw values:

$$\text{roll}' = -\tan^{-1}\left(\frac{M_{2,1}}{M_{2,2}}\right)$$

$$\text{pitch}' = \sin^{-1}\left(M_{2,0}\right)$$

$$\text{yaw}' = -\tan^{-1}\left(\frac{M_{1,0}}{M_{0,0}}\right)$$

   Note that in implementing these calculations it is important to use the ATAN2 rather than the ATAN arctangent implementation in order to retain the correct quadrants for the Euler angles. This is not a concern in Earth-view imagery where the angles are always small, but becomes an issue for these lunar/stellar ACS to ECI angles.

3. Store the ECI roll-pitch-yaw values in the model attitude data table.

At the completion of this sub-algorithm the model structure contains the image frame time stamps, the multispectral and panchromatic sample, integration, and settling times, the ancillary ephemeris data, in both ECI and ECEF representations, covering the image, and the ancillary attitude data covering the image.

Jitter Correction Data Preprocessing

Jitter correction preprocessing operates on the roll-pitch-yaw attitude data stream extracted from the spacecraft ancillary data to separate the low frequency spacecraft pointing effects from the higher frequency jitter disturbances. The low frequency pointing model is used for line-of-sight projection and other geolocation processing while the high frequency jitter effects are applied as per-line corrections during image resampling. To implement this frequency separation in the line-of-sight model the original attitude sequence is passed through a low pass filter with a cutoff frequency defined as a parameter in the CPF. This cutoff frequency will nominally be in the 1 Hz to 10 Hz range. The value ultimately selected for this cutoff frequency will depend upon the actual disturbance profile observed in the spacecraft attitude data. The high frequency data stream should be limited in magnitude to sub-

pixel (ideally sub-half-pixel) effects, but the lower the cutoff frequency can be, the sparser (and smaller) the OLI resampling grid can be made in the line (time) dimension.

The low pass filtered version of the attitude sequence is differenced with the original data to construct the complementary high pass data sequence. The high pass sequence is then interpolated at the image line times for the OLI panchromatic band to provide a table containing high frequency roll-pitch-yaw corrections for each image line. This jitter table is stored in the OLI line-of-sight model. The original attitude sequence in the line-of-sight model is replaced with the low pass filtered sequence to avoid double counting the high frequency effects. This process is depicted in figure 4.

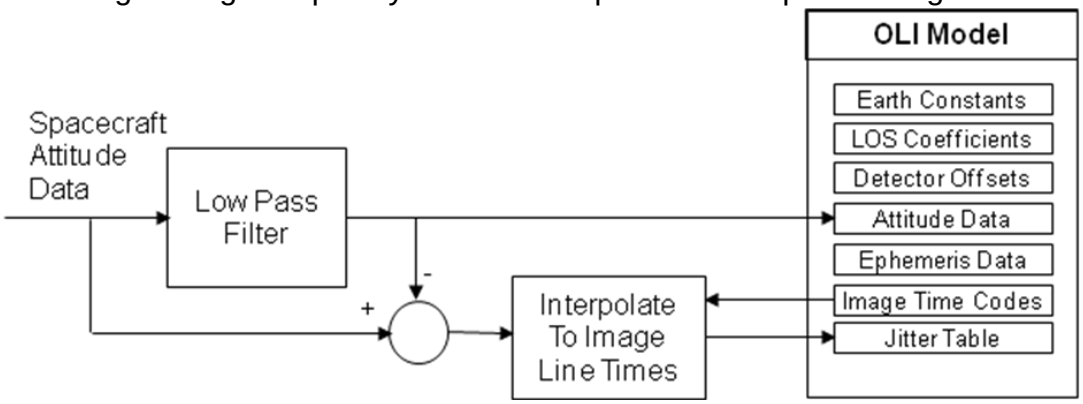


Figure 4: Jitter Correction Table Generation Data Flow

The jitter table construction processing sequence is as follows:

1. Extract a copy of the original attitude data sequence from the OLI line-of-sight model.
2. Retrieve the low pass filter cutoff frequency from the CPF.
3. Design a low pass filter with the desired cutoff frequency and apply it to the attitude data.
    a. Use the cutoff frequency and attitude data sampling time to compute the size of the desired filter as follows:
        i. Compute the normalized cutoff frequency (the ratio of the cutoff frequency to the attitude data sampling frequency):
            n_cutoff = cutoff_frequency / attitude_sample_frequency
        Note that this is the same as:
            n_cutoff = cutoff_frequency * attitude_sample_time
        ii. Compute the number of samples per cycle at the cutoff frequency:
            Nsamp = 1 / n_cutoff
        iii. Multiply the number of samples per cycle by 3 and add 1 to yield the desired filter size:
            FSize = 3*Nsamp + 1
        iv. If this results in an even filter size, add one:
            If (FSize modulo 2 == 0) FSize = FSize + 1
    b. Use the Remez exchange algorithm to design the filter and generate the filter weights. The standard Parks-McClellan finite impulse response (FIR) digital filter design method uses the Remez exchange algorithm (ref. Theory and Application of Digital Signal Processing, Rabiner and Gold, Prentice-Hall, 1975). A C implementation of this algorithm called remez.c, authored by Jake Janovetz at the University of Illinois, is available as shareware. This implementation specifies the desired (low pass, in this case) filter response using the following parameters:
        i. Filter size (number of taps) – FSize computed in item a. above.

ii. Number of frequency bands to use – 2, one pass band (low frequency) and one stop band (high frequency).
iii. Band frequency bounds – 0 to the normalized cutoff frequency (n_cutoff) for the pass band and 1.5*n_cutoff to 0.5 (normalized Nyquist frequency) for the stop band.
iv. Desired band gains – 1 for pass band (low) and 0 for stop band (high).
v. Band weights (how tightly to constrain the actual filter response to the design filter response in each band) – 1 for pass band and 10 for stop band.
vi. Filter type – BANDPASS (the low pass filter is a special case of the more general BANDPASS filter type supported by the remez algorithm.

c. Make sure the synthesized filter is normalized (weights sum to 1) by adding the filter tap values and dividing each tap by the total.

sum = Σ h[i] where h[i] are the FSize filter taps.
h'[i] = h[i] / sum for i = 1 to FSize.

d. Convolve the filter with the roll-pitch-yaw attitude data one axis at a time:

```
half_size = FSize / 2
for index = 0 to num_rpy – 1
    low_roll[index] = low_pitch[index] = low_yaw[index] = 0
    for ii = -half_size to half_size
        if ( index + ii < 0 ) j = -index – ii
        else if ( index + ii < num_rpy ) j = index + ii
        else j = 2*num_rpy – index - ii – 1
    low_roll[index] += roll[j]*h[ii+half_size]
    low_pitch[index] += pitch[j]*h[ii+half_size]
    low_yaw[index] += yaw[j]*h[ii+half_size]
```

4. Subtract the low pass filtered sequences from the original sequences to extract the high frequency portion of the data, and transfer any residual bias (non-zero mean value) from the imaging portion of the high frequency sequence to the low frequency sequence:

```
roll_bias = pitch_bias = yaw_bias = 0
att_pts = 0
for index = 0 to nrpy–1
    high_roll[index] = roll[index] – low_roll[index]
    high_pitch[index] = pitch[index] – low_pitch[index]
    high_yaw[index] = yaw[index] – low_yaw[index]
    if ( image_start_time < attitude_time[index] < image_stop_time )
        roll_bias += high_roll[index]
        pitch_bias += high_pitch[index]
        yaw_bias += high_yaw[index]
        att_pts += 1
roll_bias = roll_bias / att_pts
pitch_bias = pitch_bias / att_pts
yaw_bias = yaw_bias / att_pts
for index = 0 to nrpy-1
    high_roll[index] -= roll_bias
    low_roll[index] += roll_bias
    high_pitch[index] -= pitch_bias
    low_pitch[index] += pitch_bias
    high_yaw[index] -= yaw_bias
    low_yaw[index] += yaw_bias
```

5. Interpolate the high frequency sequence values at the panchromatic band line sampling times to create the model jitter table:

    For each panchromatic image line = 0 to number of pan lines:

        Compute the line sampling time as:

            index = line / 2

            pan_time = line_time_stamp[index] - pan_settle_time
                - pan_integration_time/2
                + (line - 2*time_index)*pan_sample_time

        Convert to time from attitude epoch:

            pan_time += image_epoch – attitude _epoch

        Interpolate high frequency roll-pitch-yaw values at this time using four point Lagrange interpolation:

            Compute starting index for interpolation:

            index = floor(pan_time / attitude_sample_time) – 1

            Compute the fractional sample offset to the pan line time:

            w = pan_time / attitude_sample_time – index – 1

            Compute the Lagrange weights:

            w1 = -w * (w – 1) * (w – 2) / 6

            w2 = (w + 1) * (w – 1) * (w – 2) / 2

            w3 = -w * (w + 1) * (w – 2) / 2

            w4 = (w + 1) * w * (w – 1) / 6

            Interpolate:

            roll = high_roll[index]*w1 + high_roll[index+1]*w2
                + high_roll[index+2]*w3 + high_roll[index+3]*w4

            pitch = high_pitch[index]*w1 + high_pitch[index+1]*w2
                + high_pitch[index+2]*w3 + high_pitch[index+3]*w4

            yaw = high_yaw[index]*w1 + high_yaw[index+1]*w2
                + high_yaw[index+2]*w3 + high_yaw[index+3]*w4

6. Replace the original model attitude data sequence with the low pass filtered attitude data sequence.

Process LOS Model Sub-Algorithm

This function loads the LOS Legendre polynomial coefficients and other model components from the CPF, and performs additional processing on the attitude and ephemeris information in the LOS model structure. It invokes the following sub-algorithms.

Read CPF Model Parameters Sub-Algorithm

This function loads model components from the CPF. In the heritage ALIAS implementation some of these model components either did not exist (e.g., instrument offset from spacecraft center of mass) or were used for image resampling but not LOS model computations (e.g., detector offset table) and so, were not included in the model. These are included in the OLI model to make it self-contained for purposes of line-of-sight computations.

Key CPF parameters loaded into the geometric model include:

1. Earth orientation parameters – the UT1UTC and pole wander (x,y) parameters for the current day are stored in the model to avoid the necessity of repeatedly looking them up in the CPF. WGS84 ellipsoid parameters (semi-major and semi-minor axes and eccentricity) are also extracted from the CPF as are physical constants such as the Earth rotation velocity and the speed of light.

2. OLI offset from spacecraft center of mass – a 3-vector that captures the small offset, in spacecraft body coordinates, between the OLI instrument, where images are captured, and the spacecraft center of mass, the position of which is reported in the ancillary ephemeris data, making it possible to translate the ephemeris data to the OLI. Technically, this would be the vector from the spacecraft center of mass to the center of the OLI entrance pupil. Note that this formulation assumes that the spacecraft on-board GPS data processing includes the GPS to spacecraft center of mass (CM) offset and that the spacecraft is, in fact, reporting CM positions not GPS antenna positions. If the ephemeris represents the GPS antenna location then we would need to know the spacecraft CM to GPS antenna offset as well.
3. OLI to attitude control system (ACS) alignment matrix – a 3-by-3 matrix that captures the relative orientation of the OLI coordinate system to the ACS coordinate system, making it possible to rotate the OLI instrument-space line-of-sight vectors into the ACS reference system. In the heritage ALIAS system this was actually represented in the CPF by an ACS to instrument rotation matrix which was inverted for each LOS model invocation. Whichever convention is used in the CPF, the LOS model should store the OLI-to-ACS rotation matrix.
4. OLI sensor parameters including the nominal detector sampling rate, integration times (pan and MS), settling times (pan and MS), and instantaneous fields of view (IFOVs), as well as the number of bands, SCAs per band, detectors per SCA, and nominal detector fill values.
5. OLI line-of-sight Legendre polynomials – a set of 6 coefficients (3 along-track and 3 across-track) for each band on each SCA. Each set of 3 forms a 2nd order Legendre polynomial that is used to evaluate a nominal LOS angle (along- or across-track) for the detectors in that band on that SCA. This is the heritage ALIAS implementation (see the Read LOS Vectors Sub-Algorithm description below).
6. OLI detector delay table – a table consisting of two values (along- and across-track) per detector reflecting the offset of each actual detector from its nominal location (as modeled by the 2nd order Legendre polynomials – see below). In the heritage ALIAS implementation these were small sub-pixel offsets that were applied in the image resampling procedure. With the OLI, this table will also contain the even/odd detector offsets as well as any offsets due to detector deselect (i.e., the operational use of a detector from one of the redundant rows). The even/odd offset had been modeled separately as a single value for each band, but the possibility of per-detector deselect offsets led to their inclusion in the per-detector offset table. This table is therefore needed in those LOS projection algorithms that utilize either actual (whole pixel offsets) or exact (full sub-pixel offsets) detector locations.

Read LOS Vectors Sub-Algorithm

This function retrieves the line of sight vectors from the CPF. The line of sight vectors are stored as sets of 2nd order Legendre polynomial coefficients. There is a unique set of 6 coefficients for each band of each SCA, 3 for the along-track polynomial and 3 for the across-track polynomial. These values are read from the CPF and stored in the LOS model. The polynomials are used to compute along- and across-track viewing angles for each nominal detector.

Initialize the Precision Model Sub-Algorithm

This function initializes the precision LOS correction model parameters. If the optional precision model input parameters are provided, those values are used. In the normal case, those parameters are absent and the correction model is initialized as follows:

Set the precision correction reference time to the beginning of the scene:

t_ref = 0.0

Set the ephemeris correction model order to zero: eph_order = 0
Set both ephemeris X correction parameters to zero:
x_corr[0] = 0.0, x_corr[1] = 0.0
Set both ephemeris Y correction parameters to zero:
y_corr[0] = 0.0, y_corr[1] = 0.0
Set both ephemeris Z correction parameters to zero:
z_corr[0] = 0.0, z_corr[1] = 0.0
Set the attitude correction model order to zero: att_order = 0
Set all three attitude roll correction parameters to zero:
roll_corr[0] = 0.0, roll_corr[1] = 0.0, roll_corr[2] = 0.0
Set all three attitude pitch correction parameters to zero:
pitch_corr[0] = 0.0, pitch_corr[1] = 0.0, pitch_corr[2] = 0.0
Set all three attitude yaw correction parameters to zero:
yaw_corr[0] = 0.0, yaw_corr[1] = 0.0, yaw_corr[2] = 0.0

Note that these parameters are used to compute the corrected ephemeris and attitude data sequences which are also stored in the model. The parameters themselves are included in the model primarily to document the magnitude of the corrections applied and to facilitate more advanced uses of the model creation logic. For example, it is sometimes useful to be able to force a particular model bias (e.g., a roll angle) into a model that is to be used for data simulation (see note 9). So, though not strictly necessary for operational data processing, these parameters aid in anomaly resolution, data simulation, and algorithm development. In normal operations, these initial correction parameters are all zero and the "corrected" attitude and ephemeris data sequences are identical to the "original" attitude and ephemeris data prior to the execution of the LOS model correction algorithm. Subsequent algorithms (e.g., LOS projection) operate on the corrected data.

Correct Attitude Sub-Algorithm
This function applies the ACS/body space attitude corrections computed by the LOS/precision correction procedure to the attitude data sequence. It outputs a parallel table of roll-pitch-yaw values with the precision corrections applied. In the model creation context the precision corrections are zero so the two sets of attitude data are identical. Though applying the precision corrections to construct the corrected attitude sequence could be said to be overkill for model creation (since the corrections are nominally zero at this point) this capability is required for LOS model correction and is used here to support the use of the model creation algorithm for data simulation and anomaly resolution as it makes it possible to force initial biases into the model. This sub-algorithm will also be used by the LOS/precision correction algorithm to create the precision model. Note that the formulation is somewhat different for Earth-view scenes (Acquisition Type = Earth) than it is for lunar and stellar observations.

Earth Scenes
For Earth-view scenes the sequence of transformations required to convert a line-of-sight in the OLI instrument coordinate system, generated using the Legendre polynomials, is:

$$\underline{\mathbf{x}}_{ECEF} = \mathbf{M}_{ORB2ECEF}\,\mathbf{M}_{ACS2ORB}\,\mathbf{M}_{Precision}\,\mathbf{M}_{OLI2ACS}\,\underline{\mathbf{x}}_{OLI}$$

where: $\underline{\mathbf{x}}_{OLI}$ is the Legendre-derived instrument LOS vector
$\mathbf{M}_{OLI2ACS}$ is the OLI to ACS alignment matrix from the CPF
$\mathbf{M}_{Precision}$ is the correction to the attitude data computed by the LOS/precision correction procedure

$\mathbf{M}_{ACS2ORB}$ is the spacecraft attitude (roll-pitch-yaw)
$\mathbf{M}_{ORB2ECEF}$ is the orbital to ECEF transformation computed using the ECEF ephemeris
$\underline{\mathbf{x}}_{ECEF}$ is the LOS vector in ECEF coordinates

Note that in the heritage ALIAS implementation the sequence was:

$$\underline{\mathbf{x}}_{ECEF} = \mathbf{M}_{ORB2ECEF}\,\mathbf{M}_{Precision}\,\mathbf{M}_{ACS2ORB}\,\mathbf{M}_{OLI2ACS}\,\underline{\mathbf{x}}_{OLI}$$

For nadir-viewing imagery the $\mathbf{M}_{ACS2ORB}$ matrix is nearly identity, so there is little difference. Since OLI will occasionally be viewing off-nadir and it is more natural to model attitude errors in the ACS/body coordinate system, the order has been reversed for LDCM. The impact is minimal in the model and LOS projection but becomes more important for the LOS/precision correction algorithm.

This new sub-algorithm pre-computes the $\mathbf{M}_{ACS2ORB}\,\mathbf{M}_{Precision}$ combination and stores the corresponding corrected roll-pitch-yaw attitude sequence in the model structure. This approach has several advantages:

1. It streamlines the application of the model for LOS projection by removing the step of explicitly applying the precision correction.
2. It allows for the use of a more complex correction model in the future since the application of the model is limited to this unit. Note that the Earth-view attitude correction model consists of the following model parameters:
   Precision reference time: t_ref in seconds from the image epoch (at the center of the image time window)
   Attitude model order: att_order = 2
   Roll bias and rate corrections: roll_corr[] = roll_bias, roll_rate
   Pitch bias and rate corrections: pitch_corr[] = pitch_bias, pitch_rate
   Yaw bias and rate corrections: yaw_corr[] = yaw_bias, yaw_rate
   This model is dealt with in more detail in the line-of-sight correction algorithm description.
3. Retaining both the original and corrected attitude sequences in the model make the model self-contained and will make it unnecessary for the LOS/precision correction algorithm to access the preprocessed ancillary data.

The disadvantage is that it doubles the size of the attitude data in the model structure.

The construction of the corrected attitude sequence proceeds as follows:

For each point in the attitude sequence j = 0 to K-1:

1. Compute the rotation matrix corresponding to the j^{th} roll-pitch-yaw values:

$$\mathbf{M}_{ACS2ORB} = \begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

2. Compute the precision correction at the time (t_att = att_seconds + att_time[j]) corresponding to the attitude sample:

   a. $$roll\_correction = \sum_{i=0}^{att\_order-1} \mathbf{roll\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$$

   b. $$pitch\_correction = \sum_{i=0}^{att\_order-1} \mathbf{pitch\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$$

c. $$yaw\_correction = \sum_{i=0}^{att\_order-1} \mathbf{yaw\_corr}[i] * (t\_att - t\_ref - image\_seconds)^i$$

Note that only the seconds of day fields are needed for the attitude and image epochs as they are constrained to be based on the same year and day.

3. Compute the rotation matrix corresponding to roll_correction (r), pitch_correction (p), and yaw_correction (y) ($\mathbf{M}_{Precision}$) using the same equations presented in step 1 above.
4. Compute the composite rotation matrix: $\mathbf{M} = \mathbf{M}_{ACS2ORB}\ \mathbf{M}_{Precision}$
5. Compute the composite roll-pitch-yaw values:

$$roll' = -\tan^{-1}\left(\frac{M_{2,1}}{M_{2,2}}\right)$$

$$pitch' = \sin^{-1}\left(M_{2,0}\right)$$

$$yaw' = -\tan^{-1}\left(\frac{M_{1,0}}{M_{0,0}}\right)$$

6. Store the composite roll'-pitch'-yaw' values in the j^{th} row of the corrected attitude data table.

Lunar and Stellar Scenes

For celestial (lunar or stellar) observations the sequence of transformations required to convert a line-of-sight in the OLI instrument coordinate system, generated using the Legendre polynomials, is:

$$\underline{\mathbf{x}}_{ECI} = \mathbf{M}_{ACS2ECI}\ \mathbf{M}_{Precision}\ \mathbf{M}_{OLI2ACS}\ \underline{\mathbf{x}}_{OLI}$$

where:
$\underline{\mathbf{x}}_{OLI}$ is the Legendre-derived instrument LOS vector
$\mathbf{M}_{OLI2ACS}$ is the OLI to ACS alignment matrix from the CPF
$\mathbf{M}_{Precision}$ is the correction to the attitude data computed by the LOS/precision correction procedure
$\mathbf{M}_{ACS2ECI}$ is the spacecraft attitude in the ECI frame derived from the ECI quaternions in the preprocessed ancillary data
$\underline{\mathbf{x}}_{ECI}$ is the LOS vector in ECI coordinates

The advantage of modeling the precision attitude corrections in ACS rather than orbital coordinates becomes apparent here, since the orbital frame is not used in the lunar case.
This sub-algorithm pre-computes the $\mathbf{M}_{ACS2ECI}\ \mathbf{M}_{Precision}$ combination and stores the corresponding corrected attitude sequence (as roll-pitch-yaw values relative to ECI) in the model structure. Another difference between the Earth-view and lunar/stellar models is in the formulation of the precision model. The lunar attitude correction model adds an acceleration term to the Earth-view correction model parameters:

Precision reference time: t_ref in seconds from the image epoch (nominally near the center of the image time window)
Attitude correction model order: att_order = 3
Roll bias, rate, and acceleration corrections: roll_corr[] = roll_bias, roll_rate, roll_acceleration
Pitch bias, rate, and acceleration corrections: pitch_corr[] = pitch_bias, pitch_rate, pitch_acceleration
Yaw bias, rate, and acceleration corrections: yaw_corr[] = yaw_bias, yaw_rate, yaw_acceleration

Due to the different orders of the Earth-view and lunar correction models, this model is stored as an array in the model structure along with a field defining the model order. The precision model is dealt with in more detail in the line-of-sight correction algorithm description.

The processing steps to construct the corrected attitude sequence is the same for lunar/stellar acquisitions, although the interpretation of the roll-pitch-yaw values is slightly different, and proceeds as follows:

For each point in the attitude sequence j = 0 to K-1:

1. Compute the rotation matrix corresponding to the j[th] ECI roll-pitch-yaw values:

$\mathbf{M}_{ACS2ECI}$ =

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

7. Compute the precision correction at the time (t_att = att_seconds + att_time[j]) corresponding to the attitude sample:

   a. $roll\_correction = \sum_{i=0}^{att\_order-1} \mathbf{roll\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$

   b. $pitch\_correction = \sum_{i=0}^{att\_order-1} \mathbf{pitch\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$

   c. $yaw\_correction = \sum_{i=0}^{att\_order-1} \mathbf{yaw\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$

   Note that only the seconds of day fields are needed for the attitude and image epochs as they are constrained to be based on the same year and day.

2. Compute the rotation matrix corresponding to roll_correction (r), pitch_correction (p), and yaw_correction (y):

$\mathbf{M}_{Precision}$ =

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

3. Compute the composite rotation matrix: $\mathbf{M} = \mathbf{M}_{ACS2ECI}\,\mathbf{M}_{Precision}$
4. Compute the composite ACS to ECI roll-pitch-yaw values:

$$roll' = -\tan^{-1}\left(\frac{M_{2,1}}{M_{2,2}}\right)$$

$$pitch' = \sin^{-1}\left(M_{2,0}\right)$$

$$yaw' = -\tan^{-1}\left(\frac{M_{1,0}}{M_{0,0}}\right)$$

   Note that in implementing these calculations it is important to use the ATAN2 rather than the ATAN arctangent implementation in order to retain the correct quadrants for the Euler angles. This is not a concern in Earth-view imagery where the angles are always small, but becomes an issue for these lunar/stellar ACS to ECI angles.

5. Store the composite roll'-pitch'-yaw' values in the j[th] row of the corrected attitude data table.

Correct Ephemeris Sub-Algorithm

The heritage ALIAS function converts the ephemeris information (position and velocity) from the Earth Centered Inertial (ECI J2000) system to the Earth Centered Earth Fixed (ECEF) system and applies the ephemeris corrections computed in the LOS/precision correction procedure to both ephemeris sets. Since both ECI and ECEF representations of the ephemeris are now provided by the ancillary data preprocessing algorithm, the first portion of the heritage algorithm is no longer necessary (or could be reused in the ancillary data preprocessing algorithm). Though applying the precision corrections to construct the corrected ephemeris sequence could be said to be overkill for model creation (since the corrections are nominally zero at this point) this capability is required for LOS model correction and is used here to support the use of the model creation algorithm for data simulation and anomaly resolution as it makes it possible to force initial biases into the model. This sub-algorithm will also be used by the LOS/precision correction algorithm to create the precision model.

The precision correction parameters are stored in the LOS model in the spacecraft orbital coordinate system as three position (x_bias, y_bias, z_bias) corrections and three velocity (x_rate, y_rate, z_rate) corrections that, like the attitude corrections, are relative to t_ref. These values must be converted to the ECEF and ECI coordinate systems. Once the precision correction is determined in the ECEF/ECI coordinate system, the ECEF/ECI ephemeris values can be updated with the precision parameters.

Loop on LOS model ephemeris points j = 0 to N-1

Compute the precision correction:

Calculate delta time for precision correction:

dtime = ephem_seconds + ephem_time[j] – t_ref – image_seconds

Calculate the change in X, Y, Z due to precision correction. Corrections are in terms of spacecraft orbital coordinates.

dx orb = model precision x_corr[0] + model precision x_corr[1] * dtime
dy orb = model precision y_corr[0] + model precision y_corr[1] * dtime
dz orb = model precision z_corr[0] + model precision z_corr[1] * dtime

where:
model precision x_corr[0] = precision (orbital) update to X position
model precision y_corr[0] = precision (orbital) update to Y position
model precision z_corr[0] = precision (orbital) update to Z position
model precision x_corr[1] = precision (orbital) update to X velocity
model precision y_corr[1] = precision (orbital) update to Y velocity
model precision z_corr[1] = precision (orbital) update to Z velocity

Construct precision position and velocity “delta” vectors.

$$[\text{dorb}] = \begin{bmatrix} \text{dx orb} \\ \text{dy orb} \\ \text{dz orb} \end{bmatrix}$$

$$[\text{dvorb}] = \begin{bmatrix} \text{model precision x\_corr}[1] \\ \text{model precision y\_corr}[1] \\ \text{model precision z\_corr}[1] \end{bmatrix}$$

Calculate the orbit to ECF transformation [ORB2ECEF] using ECEF ephemeris (See the ancillary data preprocessing ADD for this procedure).

Transform precision "delta" vectors to ECEF.

$$[\text{def}] = [\text{ORB2ECEF}][\text{dorb}]$$

$$[\text{dvef}] = [\text{ORB2ECEF}][\text{dvorb}]$$

Adjust ECEF ephemeris by the appropriate "delta" precision vector and store the new ephemeris in the model. These ephemeris points will be used when transforming an input line/sample to an output projection line/sample.

$$\text{model ef postion} = \text{ephemeris ecef postion} + \text{decf}$$

$$\text{model ef velocity} = \text{ephemeris ecef velocity} + \text{dvecf}$$

where:

All parameters are 3x1 vectors
ephemeris ecef values are the interpolated one-second ephemeris values in ECEF coordinates

Calculate the orbit to ECI transformation [ORB2ECI] using ECI ephemeris.

Transform precision "delta" vectors to ECI.

$$[\text{deci}] = [\text{ORB2ECI}][\text{dorb}]$$

$$[\text{dveci}] = [\text{ORB2ECI}][\text{dvorb}]$$

Adjust ECI ephemeris by the appropriate "delta" precision vector and store the new ephemeris in the model. These ephemeris points will be used with lunar/stellar observations.

$$\text{model eci postion} = \text{ephemeris eci postion} + \text{deci}$$

$$\text{model eci velocity} = \text{ephemeris eci velocity} + \text{dveci}$$

where:

All parameters are 3x1 vectors
ephemeris eci values are the interpolated one-second ECI ephemeris

Move Satellite Sub-Algorithm

This function computes the satellite position and velocity at a delta time from the ephemeris reference time using Lagrange interpolation. This is a utility sub-algorithm that accesses the model ephemeris data to provide the OLI position and velocity at any specified time. Since the model ephemeris arrays are inputs to this sub-algorithm it will work with either the ECI or ECEF ephemeris data.

Table 1 below summarizes the contents of the LOS model structure. The estimated size of this structure is approximately 1.5 megabytes.

LOS Model Structure Contents
Satellite Number (8)
Format Version Number (for documentation and backward compatibility)
WRS Path
WRS Row (may be fractional)
Acquisition Type (Earth, Lunar, Stellar)
Earth Parameters
UT1UTC Correction (in seconds)
Pole Wander X Correction (in arc seconds)
Pole Wander Y Correction (in arc seconds)
WGS84 Ellipsoid Semi-Major Axis (in meters)
WGS84 Ellipsoid Semi-Minor Axis (in meters)
WGS84 Ellipsoid Eccentricity (dimensionless)
Earth Angular Velocity (radians/second)
Speed of Light (meters/second)
Image Model
Number of image lines
Image UTC epoch: image_year, image_day, image_seconds
For each line: frame time offset (in seconds) from image epoch
For each line: roll, pitch, yaw high frequency jitter correction (in radians)
Nominal alignment fill table (from CPF) one value per band per SCA (in pixels)
Detector alignment fill table (from L0R/L1R) one value per detector (in pixels)
Sensor Model
OLI to ACS reference alignment matrix [3x3]
Spacecraft center of mass to OLI offset in ACS reference frame [3x1] in meters
Integration Times (MS and pan) in seconds
Computed Sample Times (MS and pan) in seconds
Detector Settling Times (MS and pan) in seconds
Number of SCAs (14)
Number of Bands (9)
Along-Track IFOVs (MS and pan) in radians
Across-Track IFOVs (MS and pan) in radians
Number of Detectors per SCA Per Band (9x1 array)
Focal plane model parameters (Legendre coefs) [NSCAxNBANDx2x3] (in radians)
Detector delay table [NSCAxNBANDx2xNDET] (in pixels)
Ephemeris Model
Scene ephemeris data UTC epoch: imgeph_year, imgeph_day, imgeph_seconds
Number of ephemeris samples
Time from epoch (one per sample, nominally 1 Hz) (in seconds)
Original ECI position estimate (X, Y, Z) (one set per sample) (in meters)
Original ECI velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec)
Original ECEF position estimate (X, Y, Z) (one set per sample) (in meters)

Original ECEF velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec)
Corrected ECI position estimate (X, Y, Z) (one set per sample) (in meters)
Corrected ECI velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec)
Corrected ECEF position estimate (X, Y, Z) (one set per sample) (in meters)
Corrected ECEF velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec)
Attitude Model
Scene attitude data UTC epoch: imgatt_year, imgatt_day, imgatt_seconds
Number of attitude samples
Time from epoch (one per sample, nominally 50 Hz) (in seconds)
Original Roll, pitch, yaw estimate (one per sample) (in radians)
Corrected Roll, pitch, yaw estimate (one per sample) (in radians)
Precision Correction Model
Precision reference time (t_ref) seconds from image epoch
Ephemeris correction order: eph_order (0 none, 2 for Earth-view and lunar/stellar)
X correction model: x_bias, x_rate (meters, meters/sec)
Y correction model: y_bias, y_rate (meters, meters/sec)
Z correction model: z_bias, z_rate (meters, meters/sec)
Attitude correction order: att_order (0 none, 2 for Earth, 3 for lunar/stellar)
Roll correction model: roll_bias, roll_rate, roll_acc (rad, rad/sec, rad/sec^2)
Pitch correction model: pitch_bias, pitch_rate, pitch_acc (rad, rad/sec, rad/sec^2)
Yaw correction model: yaw_bias, yaw_rate, yaw_acc (rad, rad/sec, rad/sec^2)

Table 1: LOS Model Structure Contents

Note that in the precision correction model only the first att_order correction model array elements are valid. For example, for Earth-view scenes att_order = 2 and roll_corr[0] = roll_bias, roll_corr[1] = roll_rate and roll_corr[2] is not used.

7.2.2 OLI Line-of-Sight Projection/Grid Generation Algorithm

7.2.2.1 Background/Introduction

The line-of-sight (LOS) projection and grid generation algorithm uses the OLI LOS model, created by the LOS model creation algorithm, to calculate the intersection of the projected lines-of-sight from selected OLI detector samples (pixels) with an Earth model (WGS84). The spacecraft position and pointing, OLI instrument alignment and offset information, and image timing data contained in the LOS model are used to construct the LOS for an individual OLI detector at a particular sample time. We then calculate the location where that line of sight intersects the Earth's surface, as defined by the WGS84 Earth ellipsoid or a specified elevation above or below that ellipsoid. LOS intersections for an array of detector samples that span each OLI SCA/FPM and spectral band are computed at the WGS84 ellipsoid surface as well as at a range of elevation levels selected to span the actual terrain elevations found in the image area. The resulting array of projected lines-of-sight forms a three-dimensional grid of input (Level 1R) image pixel line/sample to output space (Level 1G) mappings that can be used to interpolate input/output pixel mappings for intermediate points. The resulting ability to rapidly compute input/output mappings greatly facilitates image resampling.

The LOS projection and grid generation algorithm can also work in an "inertial direction" mode in which the output space is in angular units with respect to a set of reference inertial directions. This mode is used to process lunar data wherein the inertial coordinates (declination and right ascension) of the moon, computed from a planetary ephemeris, are used as the reference to define the output image frame. In this case the lines-of-sight are computed in inertial coordinates but are not projected to the Earth's surface.

Concerns about the temporal (line direction) grid density that would be required to adequately capture attitude deviations (jitter) at frequencies above 10 Hz motivated the addition of new grid functionality to support high frequency image correction at image resampling time. Specifically, jitter sensitivity coefficients were added to each grid cell to allow the high frequency attitude data in the OLI line-of-sight model jitter table to be converted to corresponding input image space line/sample offsets. These coefficients are used by the resampler to compute high frequency line/sample corrections that refine the output-to-input space image coordinate mappings provided by the grid. This allows the grid to model only lower frequency effects making a sparser grid sampling in the time (line) direction possible.

Due to layout of the OLI focal plane, there are along-track offsets between spectral bands within each SCA, along-track offsets between even and odd SCAs, and a reversal of the band ordering in adjacent SCAs. This leads to an along-track offset in the imagery coverage area for a given band between odd and even SCAs as well as an offset between bands within each SCA. To create more uniform image coverage within a geometrically corrected output product, the leading and trailing imagery associated with these offsets is trimmed (at image resampling time) based on image active area bounds stored in the grid.

7.2.2.2 Dependencies

The OLI LOS projection and grid generation algorithm assumes that the OLI LOS model creation algorithm has been executed to construct and store the OLI LOS model.

7.2.2.3 Inputs

The LOS projection and grid generation algorithm and its component sub-algorithms use the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs
ODL File (implementation)
CPF File Name
LOS Model File Name
DEM File Name
NOVAS Planetary Ephemeris File Name (for lunar processing)
Output Image Framing Parameters:
WRS Path for path-oriented scene framing (not necessarily the LOS model path)
WRS Row for path-oriented scene framing (not necessarily the LOS model row)
Map Projection (UTM, SOM, PS)
UTM Zone (use 0 to have code compute the zone)
Map Projection Parameters
Output Pixel Size(s)
Output Image Orientation
Frame Type (e.g., MINBOX)
Frame Bounds (e.g., corner coordinates, image size)
Grid Options:
Bands to Grid
CPF file contents
Thresholds and Limits (replaces System Table)
Grid Density (line/sample/height)
Default (WGS84) Spheroid parameter and Datum Codes
Scene framing band priority list
OLI LOS Model file contents (see LOS Model Creation ADD for additional detail)
WGS84 Earth Ellipsoid parameters
Earth Angular Velocity (rotation rate) in radians/second
PAN and MS settling times
Speed of light (in meters/second)
Acquisition Type (Earth, Lunar, Stellar)
OLI to ACS reference alignment matrix
Spacecraft CM to OLI offset in ACS reference frame (new)
Focal plane model parameters (Legendre coefs)
Detector delay table
Smoothed ephemeris at 1 second intervals (original and corrected)
Low pass filtered attitude history (original and corrected)
High frequency attitude perturbations (roll, pitch, yaw) per image line (jitter table)
Image time codes
Integration Time (MS and Pan)
OLI MS and pan detector settling times (msec)
Nominal detector alignment fill table
L0R detector alignment Fill Table
DEM file contents
Min and Max Elevation
NOVAS Planetary Ephemeris file contents
JPL Ephemeris Table (DE405) for celestial bodies (i.e., the moon) (see note 1)

7.2.2.4 Outputs

OLI Grid (see Tables 1 and 2 below for detailed grid structure contents)
Grid Header (WRS path/row, acquisition type)
Output Image Framing Information (corner coordinates, map projection)
Image active area latitude/longitude bounds (for each band)
Grid Structure Information (number of bands/SCAs)
Grid Structures (one per SCA, per band)
Band number
Image dimensions (line/sample)
Pixel size
Grid cell size (image lines/samples per cell)
Grid dimensions (# rows/# columns/# Z-planes)
Z-plane zero reference and height increment
Arrays of input line/sample grid point coordinates
Arrays of output line and sample grid point mappings
Arrays of even/odd offset coefficients (2 per grid cell)
Arrays of forward (input/output) mapping polynomials (8 per grid cell per Z-plane)
Arrays of inverse (output/input) mapping polynomials (8 per grid cell per Z-plane)
Arrays of roll-pitch-yaw jitter line sensitivity coefficients (3 per grid cell per Z-plane)
Arrays of roll-pitch-yaw jitter sample sensitivity coefficients (3 per grid cell per Z-plane)
Rough mapping polynomials (one set per Z-plane)

7.2.2.5 Options

A NOVAS planetary ephemeris file (JPL DE405) must be provided when the Acquisition Type (in the LOS model) is Lunar.

7.2.2.6 Prototype Code

Input to the executable is an ODL file; output is a HDF4 formatted resampling grid file.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The following text is a brief description of the main set of modules used within the prototype with each module listed along with a very short description. It should be noted that not all library modules are referenced in the explanations below. The modules within the main oligrid directory of the prototype are discussed and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

oligrid
Main driver for generating the resampling grid. Calls modules to retrieve user parameters, establish the output image frame extent, and populate the grid structure with appropriate input to output, and output to input, mapping parameters.

get_parms
This routine opens the input ODL parameter file, reads the grid parameters, closes the parameter file, and returns the parameters. Also will read the DEM, if the DEM is given as an input parameter, and determine the elevation extent within the DEM file. This elevation extent will then be used for establishing the z-plane parameters within the grid structure.

oli_get_model
Reads the OLI geometric model file and populates data within the OLI geometric model structure.

read_num_ls_l0ra
This routine extracts the number of image lines from the Level 1R image and the number of samples per band per SCA from the sensor model portion of the LOS model. The routine then returns the number of lines and samples for the input band numbers. These values, along with the grid cell size, will be used to determine grid point locations. The number of lines and samples will be returned in their respective arrays, in band-referenced order. This is similar to the manner in which the grid is stored. Thus the nlines and nsamps arrays must be of size nbands.

det_num_grid_ls
This routine will determine the number of input points to be stored in the grid according to the grid sampling rate or grid cell size chosen.

validate_utm_zone
This routine validates the UTM zone that was entered as an ODL parameter. The scene center longitude will be used for this verification. The nominal UTM zone to use is computed from the scene center longitude but the projection may be forced to an adjacent zone using input parameters. In particular, each WRS path/row may be preassigned to a UTM zone so that the same zone is always used for scenes near UTM zone boundaries. This should not introduce a zone offset greater than 1. The validation is performed by computing the UTM zone in which the scene center falls and then determining whether the input UTM zone (if any) is within one zone of the nominal zone.

oli_malloc_grid
Allocates memory for the grid based on image size and output elevation extent.

setup_jpl_solarsystem
Initializes JPL routines needed to determine position of the moon. Only used for lunar acquisitions.

calc_active_area
This routine determines the bounds of that portion of the output image frame that contains actual OLI imagery, excluding "ragged" band/SCA edges. The resulting active area bounds for each spectral band are stored in the grid for subsequent use by the image resampling logic.

north_up
This routine will determine the frame in output space for the north-up product. The actual frame is based on the output band's pixel size, but the frame is the same for every band. The method used to determine the scene corners depends on whether the corners were user input (PROJBOX) or calculated by projecting the Level 1R image corners (MAXBOX) but the framing logic is essentially the same in each case. Once given as input, or computed, the latitude/longitude scene corners are converted to the defined map projection, the extreme X and Y coordinates are found, and these extreme points are rounded to a whole multiple of the pixel size.

calc_stellar_size
Determines the output image extent for a stellar acquisition. Extent is based on SCA corners.

calc_lunar_size

Determines the output image extent for a lunar acquisition. Extent is based on either all of the SCA corners for all bands or only the SCA that contains the moon.

point_in_polygon
Simple point in a polygon check. Used with lunar process for determining if the moon lies within a SCA.

oli_moonpos_ls
Given a Level 1R line and sample location this module calculates the relative line of sight between the moon and satellite sensor.

oli_moonpos
Given a Julian day, this routine calculates the moon's position. Calls the JPL NOVAS libraries to determine the moon's position. Coordinates are given in terms of ECI true-of-date.

maxbox
This routine determines the frame in output space for the maxbox north-up product. Image framing is based on maximum image extent derived from SCA corners.

path_oriented
This routine will provide a path-oriented projection that is framed to a nominal WRS scene. The user specifies only the projection, pixel size, and the path and row of the scene.

det_grid_ls
Given the number of grid lines and samples that will be sampled in the input imagery, this routine calculates where each grid cell point will fall in the input Level 1R image. These grid cell points will fall at integer locations in the input imagery.

exx_mapedg
This routine calculates the minimum and maximum projection coordinates for given upper left and lower right latitude, longitude coordinates.

pad_corners
This routine pads the input corners by a defined factor of the pixel size. The x/y min and max values are input for the corner locations. These values are padded by PADVAL * the pixel size.

calc_center_and_rotation_angle
This routine will return the scene center and rotation angle for a nominal WRS scene. The WRS path and row of the input scene and the projection parameters are needed as input. Note: The WRS_Lat and WRS_Long are the Center_Lat_Long that need to be returned from this routine. The Heading angle is the WRS rotation angle, i.e., the image orientation relative to geodetic north.

calc_path_oriented_frame
Given the center point and rotation angle, this function will calculate the image corner coordinates in an SOM or UTM product. It also calculates the first-order polynomial coefficients which map output line/sample coordinates to their corresponding output projection coordinates. This routine will determine the frame in output space for the path-oriented product. The frame is calculated for each band, but the frame must be the same for every band.

angle_to_map
This routine will convert the WRS rotation angle (from geodetic north) to a frame orientation angle in map coordinates. The orientation angle will be retained in the grid structure.

path_maxmin_box
This routine will provide a path-oriented product whose frame is large enough to contain all bands (maxbox).

calc_path_oriented_maxbox_frame
This routine calculates the path-oriented frame for the maxbox approach.

make_grid
This routine establishes the input to output mappings. It invokes make_grid_point for each point to compute the mapping, and then invokes make_grid_sensitivity for each point to compute the jitter sensitivity coefficients.

make_grid_point
Calculates the input to output space mapping for a single grid point. Calls oli_forward_model to perform input space location to output space location mappings.

make_grid_sensitivity
Calculates the roll-pitch-yaw to input space line/sample jitter sensitivity coefficients for one grid point. Calls oli_forward_model_pert while varying the spacecraft attitude, the input space line number, and input space sample number to determine the corresponding output space sensitivity. It then finds the input space offsets that provide the same effect in output space as a given attitude perturbation, yielding the input space correction needed to compensate for a unit jitter disturbance for each spacecraft axis.

oli_init_lunar_projtran
Initializes the position of the moon with respect the lunar acquisition. Needed for oli_lunar_projtran.

oli_forward_model
For a given a Level 1R line, sample, band and SCA location, propagates the forward (geometric) model to determine a latitude and longitude for the specified point.

oli_forward_model_pert
A variant of oli_forward_model that accepts an additional input roll-pitch-yaw attitude perturbation array. This perturbation is added to the spacecraft attitude interpolated from the OLI LOS model at the time corresponding to the input space line/sample point being projected. This capability is used by make_grid_sensitivity in determining the jitter sensitivity coefficients.

oli_findtime
This function finds the time into the scene given the Level 1R line, sample, and band. The input sample number is 0-relative and relative to the SCA.

oli_findlos
This function finds the line of sight vector in sensor coordinates, using the Legendre polynomial LOS model stored in the LOS model.

oli_findatt
This function computes the attitude, or roll, pitch, yaw, for a given time.

oli_findjit
This function is invoked by oli_forward_model when the input detector type parameter is set to EXACT. This is currently only used by the OLI L0Rp data simulator. This unit uses the input time to extract the high frequency attitude correction from the jitter table in the OLI LOS model, so that it can be added to the low frequency spacecraft attitude result in oli_forward_model. This unit is not invoked by grid generation processing, where the detector type is NOMINAL, but as part of the forward line-of-sight model, it is described here for completeness.

l8_movesat
This function computes the satellite position and velocity at a delta time from the ephemeris reference time using Lagrange interpolation.

l8_attitude
This function finds the line of sight vector from the spacecraft to a point on the ground by transforming the line of sight vector in sensor coordinates to perturbed spacecraft coordinates.

geo_center_mass_corr
Adjusts the observation vector according to the spacecraft center of mass.

geo_corr_vel_aberr
Adjusts line of sight vector for velocity aberration.

geo_findtarpos
This function finds the position where the line of sight vector intersects the Earth's surface. Used only for Earth based acquisitions.

geo_corr_light_travel_time
Adjusts target location according to the light travel time. Used only for Earth based acquisitions.

geo_centh2det
This function converts between geocentric and geodetic coordinates. Used only for Earth based acquisitions.

exx_cart2sph
Convert between cartesian and spherical coordinates. For grid generation, applies only towards stellar and lunar acquisitions.

exx_projtran
This function converts coordinates from one map projection to another. The transformation from geodetic coordinates to the output map projection depends on the type of projection selected. The mathematics for the forward and inverse transformations for the Universal Transverse Mercator (UTM), Polar Stereo Graphic, and the Space Oblique Mercator (SOM) map projections are handled by U.S Geological Survey's (USGS) General Cartographic Transformation Package (GCTP), which may be obtained at http://edcftp.cr.usgs.gov/pub/software/gctpc/.

oli_lunar_projtran

Calculates the output line and sample location given the right ascension and declination angles associated with the sensor line-of-sight vector of a lunar acquisition. Serves as the equivalent exx_projtran for a lunar based acquisition.

exx_proj_err
This function reports projection transformation package errors. The function receives a GCTP error code and prints the correct error message.

gctp
Map projections are handled by U.S Geological Survey's (USGS) General Cartographic Transformation Package (GCTP), which may be obtained at http://edcftp.cr.usgs.gov/pub/software/gctpc/.

xxx_eval
Applies a polynomial at a given point.

calc_map_coefs
This routine calculates the bilinear mapping coefficients for each grid cell. Coefficients are calculated for mapping from input location to output location (forward mapping) and for mapping from output location to input location (inverse mapping). A separate mapping function is used for lines and samples. This equates to four mapping functions. A set of four mapping functions is calculated for each grid cell, for each SCA, for every band, and for every elevation plane that is stored in the grid.

exx_calc_forward_mappings
This function, given grid points in both input and output space, uses the Calculate Map Coefficients algorithm described in the Procedure section to generate the mapping polynomial coefficients needed to convert from a line/sample in input space (satellite) to one in output space (projection). It generates these coefficients for every cell in the grid.

exx_calc_inverse_mappings
This function, given grid points in both input and output space, uses the Calculate Map Coefficients algorithm described in the Procedure section to generate the mapping polynomial coefficients needed to convert from a line/sample in output space (projection) to one in input space (satellite). It generates these coefficients for every cell in the grid.

calc_rough_map_coefs
This routine will find the rough mapping coefficients for the grid.

oli_grid_cell_poly
This utility function calculates a "rough" mapping of output to input lines/samples. The coefficients returned from this function are used as a rough estimate of an inverse model.

calc_det_offsets
This function computes the detector offset values and stores linear mapping coefficients associated with detector offsets in the grid structure.

oli_all_ols2ils

This utility routine maps an output space line/sample back into its corresponding input space line/sample. This is done using the "rough" polynomial from the grid to determine an initial guess at an input space line and sample. From this initial guess a grid cell row and column is calculated and the inverse coefficients for that cell are retrieved from the grid. These coefficients are used to determine an exact input space line and sample (in extended space).

oli_findgridcell

This utility function finds the correct grid cell that contains the output line/sample location. It finds the correct grid cell containing the output pixel by first determining the set of grid cells to be checked. It then calls a routine to perform a "point in polygon" test on each of these grid cells to determine if the pixel does indeed fall within that grid cell.

7.2.2.7 Procedure

The LOS Projection algorithm uses the geometric LOS model created by the LOS Model Creation algorithm to relate OLI image pixels to ground locations or, in the case of lunar/stellar images, to ECI directions. The LOS model contains several components including: Earth orientation parameters, an image model (validated image time codes), a sensor model, an ephemeris model, and an attitude model. The Level 1R image line/sample location is used to compute a time of observation (from the image model), a LOS vector (from the sensor model), the spacecraft position (from the ephemeris model) at the time of observation, and the spacecraft attitude (from the attitude model) at the time of observation. The LOS vector is projected to the Earth's surface, either the topographic surface at a specified elevation (e.g., derived from an input Digital Elevation Model), or the WGS84 ellipsoid surface, to compute the ground position associated with that Level 1R image location. This LOS projection procedure relating an input image location to an output ground location is referred to as the forward model. In image resampling, we typically need to find the Level 1R input space line/sample location corresponding to a particular Level 1G output space location so that the corresponding image intensity can be interpolated from the Level 1R data. This "inverse model" computation must be performed for every pixel in the output Level 1G product. To make this computation efficient, we create a table, or grid, of input/output mappings, parameterized by height, for use by the image resampling algorithm. Both the forward model and grid generation procedures are described in this algorithm description document.

7.2.2.7.1 The Geometric Grid

The geometric grid provides a mapping from input Level 1R line/sample space to output Level 1G line/sample space. As such, it incorporates not only the sensor LOS to Earth intersection geometry captured by the forward model, but also the output image framing information, such as scene corners, map projection, pixel size, image orientation, and the bounds of the active image area for each band. The gridding procedure generates a mapping grid that defines a transformation from the instrument perspective (input space) to a user specified output projection on the ground (output space). This output frame may be map-oriented (north-up) or path-oriented for Earth-view acquisitions. Celestial (lunar/stellar) acquisitions use an output frame based on inertial right ascension and declination coordinates. Once the frame is determined in output space, the input space is gridded. Then the grid in input space is mapped to the output space using the forward model. Transformation coefficients to transform a grid cell from input to output space are determined, as well as coefficients to transform a grid cell from output to input space.

The concept behind creating this resampling grid is to define only a sparse set of points for the relationship between an input line and sample location to output line and sample location (see Figure

1). Four grid points define a grid cell. A grid cell is defined as a rectangle in input space but will be distorted when mapped to the output space. The sampling of points between grid cell points is chosen such that any two points defining a grid cell and a line in input space will map to a line in output space. Therefore every grid cell defines a bilinear mapping between the input and output space and vice versa. The method of only mapping and storing a small set of input points is much more efficient than trying to map points individually by invoking the LOS model for each point. This is especially the case since a rigorous implementation of the inverse model would have to be iterative.

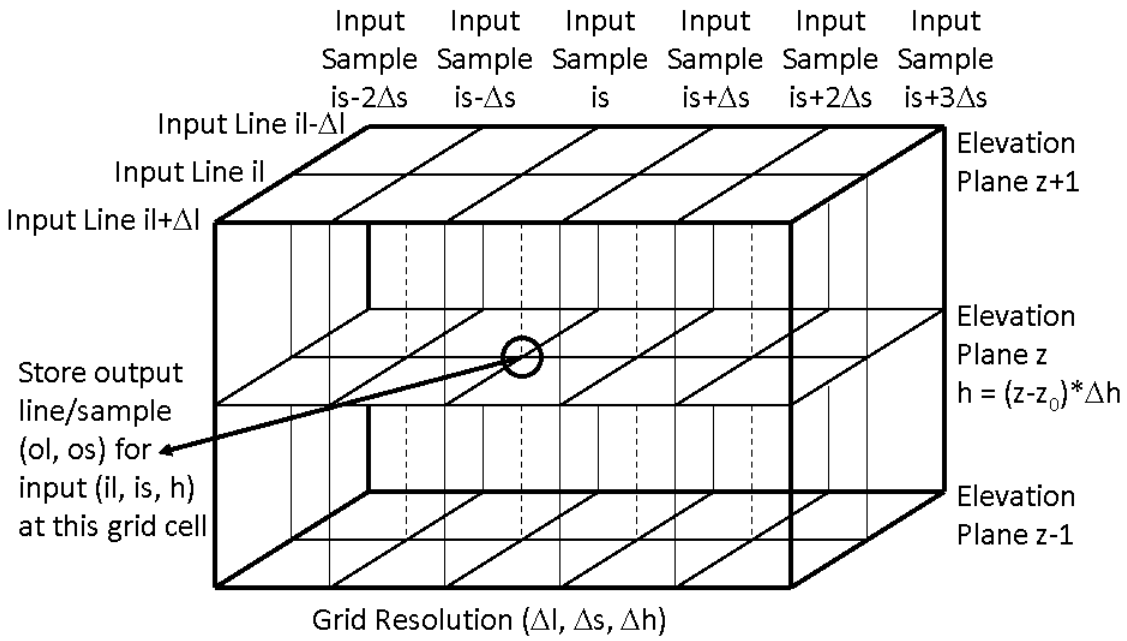


Figure 1: The 3D grid structure stores the output space line/sample coordinates corresponding to an array of input space line/sample/height coordinates.

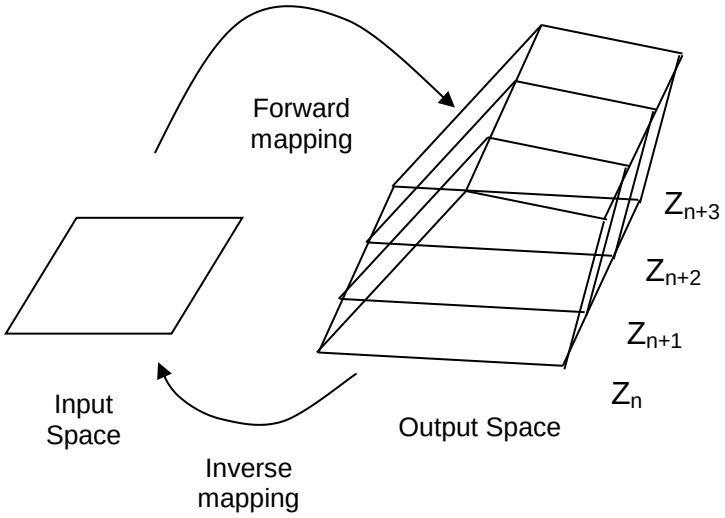


Figure 2: Forward and Inverse Mapping Using the Grid

The LOS projection grid contains projection information and three groups of mapping coefficients—one for mapping each grid cell from output space to input space (inverse), a second for mapping each

grid cell from input space to output space (forward), and third that gives an approximation or "rough" mapping of output space to input space. The first two mappings are described by a set of bilinear polynomials. The input space is represented by a line and sample location while the output space is represented by a line and sample location along with a Z component, where Z represents elevation. The output lines and samples can in turn be converted to X, Y projection space location by using the output image's upper left projection coordinate and pixel size information in the grid header. Figure 2 shows how one input grid cell is mapped to a number of output grid cells, each grid cell representing a different elevation.

The number of grid cells is dependent on the line and sample size of each grid cell in the input image, elevation maximum, elevation minimum, and elevation increment. The input space is made up of evenly spaced samples and lines, values are associated with integer locations and can be indexed by an array of values: input_line[row] and input_sample[column]. Row refers to the index number, or row number, associated with the line spacing while column refers to the index number, or column number, associated with the sample spacing. The output lines and samples typically do not fall on integer values (see Figure 3). This creates a two dimensional array of indices for output line and sample locations. Adding elevation indices produces a three dimensional array for output line and sample locations. The output lines and samples are then indexed by output_line[z][row][column] and output_sample[z][row][column] where Z refers to an elevation value. The row and column are the indices associated with the gridding of the raw input space. Since there is a mapping polynomial for each grid cell, the mapping polynomial coefficients are indexed by the same method as that used for output lines and samples; i.e. there are z*row*column sets of mapping coefficients.

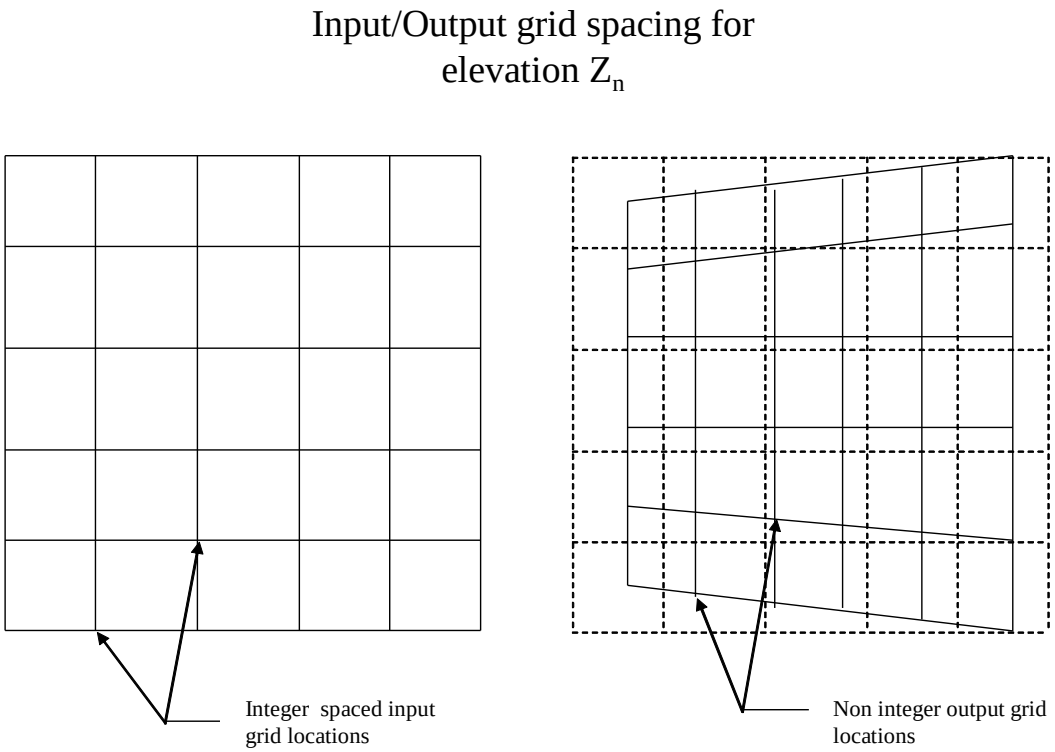


Figure 3: Mapping integer locations to "non-integer" locations

If a grid is being generated for a non-terrain corrected image (i.e., no correction for relief is being applied) then the index for z is set such that $z_{elev=0}$ = zero elevation. Note that $z_{elev=0}$ does not necessarily have to be the first index in the array since there could be values for negative elevations. If the grid is being generated for a terrain corrected image, then the indexes z_n and z_{n+1} are used such that the elevation belonging to the output location falls between the elevations associated with the indexes n and n+1. When performing an inverse mapping for a terrain corrected image, two sets of input lines and samples are calculated from the polynomials for n and n+1. The actual input line and sample is interpolated between these lines and samples.

Example:

Output line/sample has r = row, c = col and z=n, n+1. If the inverse mapping coefficients are *a* and *b* for line and sample respectively then:

$input\_line_n = bilinear(a_n, output\_line, output\_sample)$
$input\_sample_n = bilinear(b_n, output\_line, output\_sample)$
$input\_line_{n+1} = bilinear(a_{n+1}, output\_line, output\_sample)$
$input\_sample_{n+1} = bilinear(b_{n+1}, output\_line, output\_sample)$

bilinear is the bilinear mapping function (described below) for each grid cell.

If *e* is the elevation for the output line and sample location then the weights used to interpolate between the two input line/sample locations are:

$$w_n = \frac{e_{n+1} - e}{e_{n+1} - e_n} \qquad w_{n+1} = \frac{e - e_n}{e_{n+1} - e_n}$$

e_n, e_{n+1} and *e* are the elevations associated with z_n , z_{n+1} , and the output line and sample respectively.

The final line/sample location is found from:

$input\_line = w_n * input\_line_n + w_{n+1} * input\_line_{n+1}$
$input\_sample = w_n * input\_sample_n + w_{n+1} * input\_sample_{n+1}$

The grid must contain a zero elevation plane. If the input minimum elevation is greater than zero it is set to zero. If the input maximum elevation is less than zero it is set to zero.

Given the elevation maximum, minimum, and increment determine the number of z planes and the index of the zero elevation plane. Adjust the minimum and maximum elevations to be consistent with the elevation increment.

The number of z planes is determined from:

$$\text{number of z planes} = (\text{int})\left(\text{ceil}\left(\frac{\text{elevation maximum}}{\text{elevation increment}} \right) - \text{floor}\left(\frac{\text{elevation minimum}}{\text{elevation increment}} \right) \right) + 1$$

The plane for an elevation of zero is then found at:

$$z_{elev=0} = -\text{floor}\left(\frac{\text{elevation minimum}}{\text{elevation increment}}\right)$$

The new minimum and maximum elevation due to the values calculated above are:

$$\text{elevation minimum} = -z_{elev=0} * (\text{elevation increment})$$
$$\text{elevation maximum} = (\text{number of } z - 1 - z_{elev=0}) * \text{elevation increment}$$

7.2.2.7.2 LOS Projection/Grid Generation Procedure Overview

The LOS Projection/Grid Generation procedure is executed in five stages:

1. Data Input - First, the required inputs are loaded. This includes reading the processing parameters from the input ODL parameter file, loading the LOS model from its HDF file, reading static gridding parameters from the CPF, and loading the elevation data from the DEM.
2. Scene Framing - The parameters of the output image space are computed based on the scene framing scheme specified in the input ODL file. This includes calculating bounds for the active image area that excludes the leading and trailing SCA imagery, and using one of several available methods for determining the Level 1G scene corners. The scene framing parameters are stored in the grid structure for eventual inclusion in the geometric metadata for the Level 1G product.
3. Grid Definition - The grid parameters are established to ensure adequate density in the space (sample), time (line), and elevation (z-plane) dimensions. The required data structures are allocated and initialized.
4. Grid Construction - The forward model is invoked for each grid intersection to construct the array of input space to output space mappings. A separate grid structure is created for each SCA and each band. The grid mapping polynomial coefficients are computed from the input space to output space mapping results for each grid cell. Once the basic grid mappings are defined, the forward model is invoked with small attitude perturbations about each axis in order to evaluate the sensitivity of the input space to output space mapping to small attitude deviations. The resulting sensitivity coefficients are stored with each grid cell for subsequent use in computing high frequency jitter corrections during image resampling. Figure 4 shows a high level data flow for the creation and use of these new coefficients.
5. Finalize and Output Grid - Derived grid parameters such as the global rough mapping coefficients, are added to the grid structure, and the entire structure is written to a disk file. This also includes evaluating the small, but significant, parallax effects caused by the time delay between when adjacent even and odd detectors sample the same along-track location. These effects are modeled in the grid as along- and across-track sensitivity coefficients that are scaled by the output point elevation and the even/odd detector offset, which can vary by pixel for OLI (due to detector deselect) rather than by band.

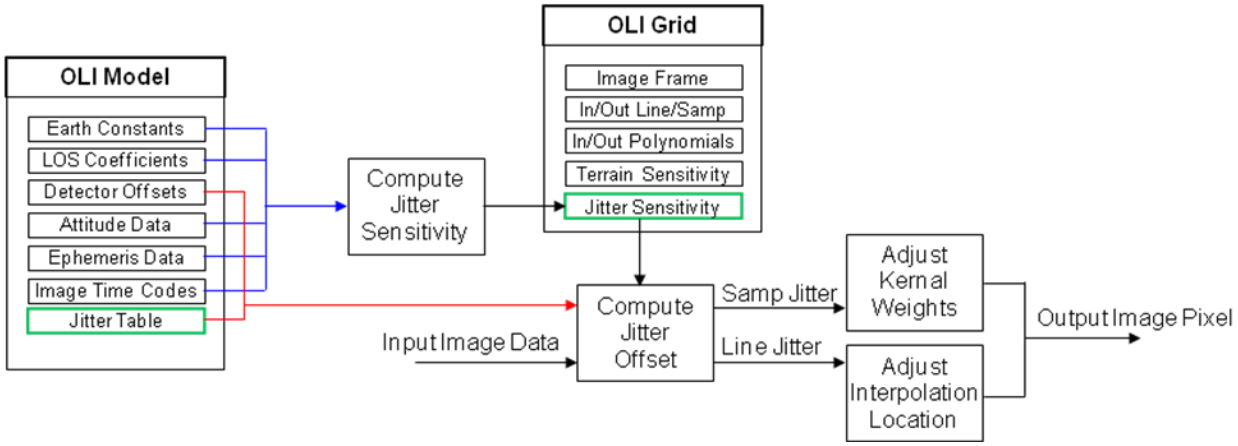


Figure 4: Jitter Correction Data Flow

Figure 5 shows a block diagram for the LOS Projection algorithm.

7.2.2.7.2.1 Stage 1 - Data Input

The data input stage involves loading the information required to perform grid processing. This includes reading the framing parameters for the output scene from the ODL file, reading grid structural parameters from the CPF, loading the LOS model structure in preparation for invoking the forward model, and reading the DEM to determine the elevation range for the image.

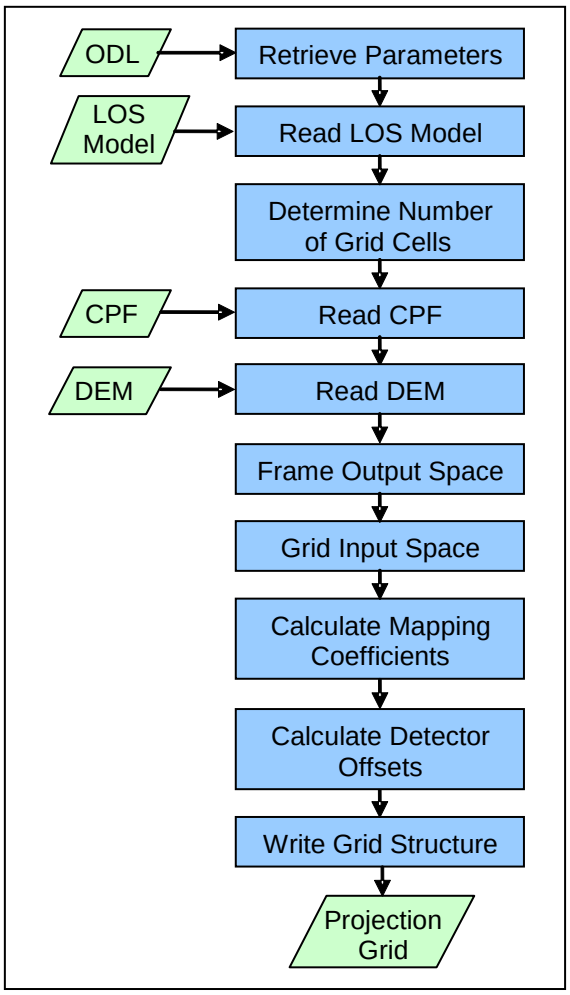


Figure 5: Line-of-Sight Projection Block Diagram

7.2.2.7.2.2 Stage 2 - Scene Framing

Framing the output image space involves determining the geographic extent of the output image to be generated by the resampler. This geographic extent of the output image space is referred to as the output space "frame," and is specified in output image projection coordinates. There are four different methods that are used to determine the output frame for Earth-viewing acquisitions. Scene framing for lunar and stellar scenes uses either a maximum bounding rectangle (maxbox) or a minimum bounding rectangle (minbox) approach using inertial LOS declination and right ascension coordinates, and is discussed separately. These methods use the calculated coverage bounds of each band/SCA in different ways, with some excluding the leading and trailing SCA imagery based on a calculated active image area, and some including the leading/trailing imagery so as to preserve all available input pixels (e.g., for calibration purposes). Thus, the calculation of the active image area for each band is the first step in scene framing.

7.2.2.7.2.2.1 Calculating the Active Image Area

The along-track offsets between spectral bands and even/odd SCAs create an uneven coverage pattern when projected into output image space. In order to provide a more regular output image coverage boundary, we define a rectangular active image area that excludes the excess trailing imagery from even SCAs and the excess leading imagery from odd SCAs. This active area is used for the minbox framing methods which seek to limit the output product area to provide consistent,

contiguous coverage, but are ignored for maxbox framing methods, where all available imagery is desired.

The active image area is computed by constructing 8 critical SCA corner points, labeled C1 through C8 in the figure below. Points C1 and C2 define the top edge of the active area, C3 and C4 the right edge, C5 and C6 the bottom edge, and C7 and C8 the left edge. Note that points C1 and C8 are the same (the upper left corner of SCA01) as are points C4 and C5 (the lower right corner of SCA14). The forward model projects these 8 line/sample locations to object space, computing the latitude/longitude coordinates of the WGS84 ellipsoid intersection for each point.

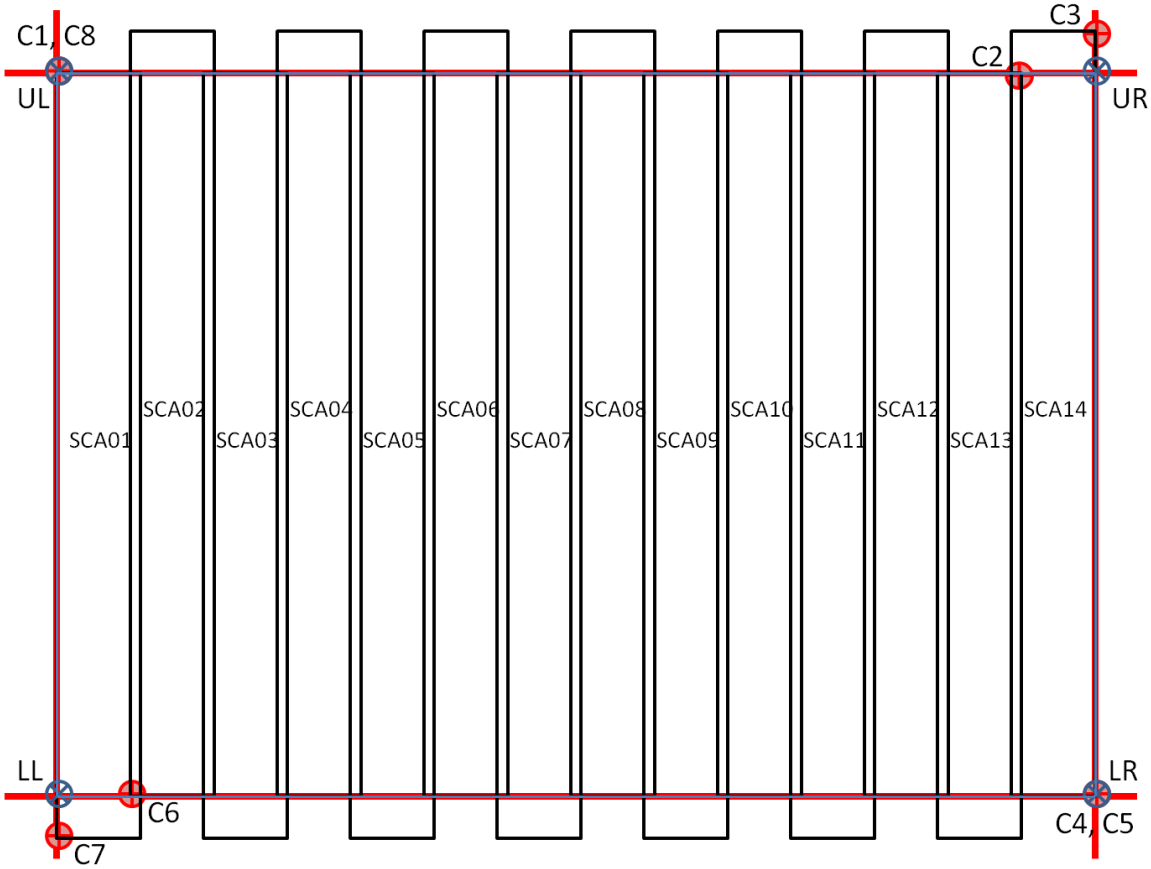


Figure 6: Active Image Area Construction

The SCA and corner point assignments are made automatically by examining the SCA across-track and along-track Legendre coefficients to determine: 1) whether SCA01 is on the left (+Y) or right (-Y) side of the scene; 2) whether even or odd SCAs lead; and 3) whether the sample number increases in the –Y or +Y direction. If the across-track Legendre constant term (coef_y0) for SCA01 is positive then it is the left-most SCA and SCA14 is the right-most. If the along-track Legendre constant term (coef_x0) for SCA01 is greater than that for SCA02, then the odd SCAs lead. If the across-track Legendre linear term (coef_y1) for SCA01 is negative, then the sample number increases in the –Y direction.

Having determined the orientation of the SCAs, we assign the top edge of the active area to the left-most leading SCA upper left (UL) corner and the right-most leading SCA upper right (UR) corner, the right edge to the right-most SCA UR and lower right (LR) corners, the bottom edge to the right-most

trailing SCA LR corner and left-most trailing SCA lower left (LL) corner, and the left edge to the left-most SCA LL and UL corners. As shown in the figure, for the OLI: C1 = SCA01 (left-most odd SCA) UL, C2 = SCA13 (right-most odd SCA) UR, C3 = SCA14 (right-most SCA) UR, C4 = SCA14 (right-most SCA) LR, C5 = SCA14 (right-most even SCA) LR, C6 = SCA02 (left-most even SCA) LL, C7 = SCA01 (left-most SCA) LL, and C8 = SCA01 (left-most SCA) UL.

The geodetic latitudes computed by the forward model are converted to geocentric longitudes using:

$\theta = \arctan((1-e^2) \tan(\phi))$

where: θ = geocentric latitude

ϕ = geodetic latitude

e^2 = WGS84 ellipsoid eccentricity squared

This creates a set of 8 geocentric latitude/longitude (θ_i, λ_i) pairs, one for each "critical" corner, noting that geocentric longitude is equal to geodetic longitude.

Use the geocentric latitude/longitude to construct a geocentric unit vector for each corner:

$$X_i = \begin{bmatrix} \cos(\lambda_i)\cos(\theta_i) \\ \sin(\lambda_i)\cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix}$$

Note that these vectors are inherently normalized.

Construct vectors normal to the top, right, bottom, and left edge great circles by taking cross products of the corner vectors:

$$X_T = \frac{X_1 \times X_2}{|X_1 \times X_2|} \qquad X_R = \frac{X_3 \times X_4}{|X_3 \times X_4|} \qquad X_B = \frac{X_5 \times X_6}{|X_5 \times X_6|} \qquad X_L = \frac{X_7 \times X_8}{|X_7 \times X_8|}$$

Construct corner vectors from the edge vectors:

$$X_{UL} = \frac{X_T \times X_L}{|X_T \times X_L|} \qquad X_{UR} = \frac{X_R \times X_T}{|X_R \times X_T|} \qquad X_{LL} = \frac{X_L \times X_B}{|X_L \times X_B|} \qquad X_{LR} = \frac{X_B \times X_R}{|X_B \times X_R|}$$

The top and bottom edges are next checked against all of the SCA corners to ensure that any curvature in the SCA field angle pattern is accounted for. This is done to suppress residual SCA edge "raggedness".

Adjust the top edge:

Construct a vector in the plane of the top edge great circle:

$$X_g = \frac{(X_{UR} - X_{UL}) \times X_T}{|(X_{UR} - X_{UL}) \times X_T|}$$

Initialize the minimum "out of plane" distance: $a_{min} = 1$

For each SCA:

For the two upper corners: UL (0,0) and UR (ns-1,0):

Use the forward model to project the corner.

Convert the geodetic latitude to geocentric latitude as above.

Construct a geocentric unit vector, X_i, as above.

Project the unit vector onto the X_g and X_T vectors and compute the ratio:

$$a_i = \frac{X_i \bullet X_T}{X_i \bullet X_g}$$

If $a_i < a_{min}$
$a_{min} = a_i$
$X_{min} = X_i$
Next corner
Next SCA

If $a_{min} < 0$ then the innermost corner lies inside the current active area and we need to adjust the top edge:

$$X'_g = \frac{(X_{\min} \bullet X_T)X_T + (X_{\min} \bullet X_g)X_g}{\left|(X_{\min} \bullet X_T)X_T + (X_{\min} \bullet X_g)X_g\right|}$$

$$X'_T = \frac{X'_g \times (X_{UR} - X_{UL})}{\left|X'_g \times (X_{UR} - X_{UL})\right|}$$

And update the top corner vectors using the adjusted edge vectors:

$$X_{UL} = \frac{X'_T \times X_L}{\left|X'_T \times X_L\right|} \qquad X_{UR} = \frac{X_R \times X'_T}{\left|X_R \times X'_T\right|}$$

Adjust the bottom edge:
Construct a vector in the plane of the bottom edge great circle:

$$X_g = \frac{(X_{LL} - X_{LR}) \times X_B}{\left|(X_{LL} - X_{LR}) \times X_B\right|}$$

Initialize the minimum "out of plane" distance: $a_{min} = 1$
For each SCA:
For the two lower corners: LL (0,nl-1) and LR (ns-1,nl-1):
Use the forward model to project the corner.
Convert the geodetic latitude to geocentric latitude as above.
Construct a geocentric unit vector, X_i, as above.
Project the unit vector onto the X_g and X_B vectors and compute the ratio:

$$a_i = \frac{X_i \bullet X_B}{X_i \bullet X_g}$$

If $a_i < a_{min}$
$a_{min} = a_i$
$X_{min} = X_i$
Next corner
Next SCA

If $a_{min} < 0$ then the innermost corner lies inside the current active area and we need to adjust the bottom edge:

$$X'_g = \frac{(X_{\min} \bullet X_B)X_B + (X_{\min} \bullet X_g)X_g}{\left|(X_{\min} \bullet X_B)X_B + (X_{\min} \bullet X_g)X_g\right|}$$

$$X'_B = \frac{X'_g \times (X_{LL} - X_{LR})}{\left|X'_g \times (X_{LL} - X_{LR})\right|}$$

And update the bottom corner vectors using the adjusted edge vectors:

$$X_{LL} = \frac{X_L \times X'_B}{\left|X_L \times X'_B\right|} \qquad X_{LR} = \frac{X'_B \times X_R}{\left|X'_B \times X_R\right|}$$

Convert the four corner vectors to the corresponding geodetic latitude/longitude:

λ = atan2(X.y, X.x)

θ = atan2(X.z, $\sqrt{X.x^2 + X.y^2}$)

ϕ = atan(tan(θ) / (1-e^2))

The four latitude/longitude corners are the bounds of the active image area.

Once the active image area bounds are calculated, the output product frame is determined using one of the following methods:

Method 1: PROJBOX

The user defines the upper-left and lower-right corner coordinates of the area of interest in target map projection coordinates. These coordinates are then projected to the output projection coordinate system using the Projection Transformation Package (see the Projection Transformation sub-algorithm below). This usually results in a non-rectangular area so a minimum-bounding rectangle is found (in terms of minimum and maximum X and Y projection coordinates) in the resulting output space. This minimum-bounding rectangle defines the output space frame. The output image pixel size is then applied to the projection space to determine the number of lines and samples in the output space. This creates an output image that is map projection north-up.

Method 2: MINBOX

The image active areas for each band, calculated previously, are converted to the specified output map projection coordinate system and used in a minimum bounding rectangle computation to create an output image frame that includes the active area for each band. The computed (latitude/longitude) active area corners are maintained in the grid for subsequent use by the image resampler, so that the output product image will not include leading/trailing SCA imagery.

Method 3: MAXBOX

The four corners of each SCA in each band are projected to the Earth. The maximum and minimum latitude and longitude found across all SCAs and all bands are used to establish the output scene frame in the manner described above for the PROJBOX method. This creates an output frame that contains all input pixels from all bands. The previously calculated image active areas are ignored in this process, and the band active area corners are all set equal to

the output product corners. Leading and trailing SCA imagery is thereby not excluded from MAXBOX framed products.

Method 4: PATH

The user specifies a path oriented Landsat product in either the SOM or UTM projection. In this case, the framing coordinates are not user-specified. The standard path-oriented frame is a preset number of lines and samples based on the Landsat WRS scene size and the maximum rotation needed to create a path-oriented product. Additional options exist to apply either MINBOX or MAXBOX logic in determining the path oriented product frame.

Method 5: LUNAR

Lunar image framing applies either the same framing methodology as MAXBOX, defining the maximum and minimum corners in right ascension and declination angles with respect to the ECI coordinate system determined by the corners of all the SCAs for all bands, or with a similar framing methodology as MINBOX, determining the corners based solely on the SCA that contains the moon. The right ascension and declination angles are adjusted according the change in orbit of the moon during image acquisition.

Method 6: STELLAR

Stellar image framing applies the same framing methodology as MAXBOX only the output space frame defining the maximum and minimum corners are in right ascension and declination angles with respect to the ECI coordinate system.

The scene framing logic uses the following sub-algorithms/routines:

a)Validate UTM Zone

The nominal UTM zone to use is computed from the scene center longitude but the projection may be forced to an adjacent zone using input parameters. In particular, each WRS path/row may be preassigned to a UTM zone so that the same zone is always used for scenes near UTM zone boundaries. This should not introduce a zone offset greater than 1. The validation is performed by computing the UTM zone in which the scene center falls and then determining whether the input UTM zone (if any) is within one zone of the nominal zone.

Shift the scene center longitude to put it in the range 0-360 degrees:
SC_long = mod(SC_long + 540, 360)

where: SC_long is the scene center longitude in degrees

Compute the nominal UTM zone (note that UTM zones are six degrees wide):
SC_zone = (int)floor(SC_long/6) + 1

See if the input zone is within one zone of the nominal zone:
if (abs(input_zone - SC_zone) < 2 or (60 - abs(input_zone - SC_zone)) < 2)
then input_zone is valid.

b) North Up Framing

Determine the scene corners. Scene corners depend on whether the corners were user input (PROJBOX) or calculated by projecting the Level 1R image corners (MAXBOX) but the framing logic is essentially the same in each case. Once given as input or computed, the latitude/longitude scene corners are converted to the defined map projection, the extreme X and Y coordinates are found, and

these extreme points are rounded to a whole multiple of the pixel size. The north-up framing methods are each described in the following sub-algorithms.

b).1. Map Edge/PROJBOX Framing
Calculates the minimum and maximum projection coordinates for given upper left and lower right latitude, longitude coordinates.

- Calculate min/max coordinates along east edge of output area by computing latitude/longitude to map x/y projections for a series of points from (minimum latitude, maximum longitude) to (maximum latitude, maximum longitude).
- Calculate min/max coordinates along west edge of output area by computing latitude/longitude to map x/y projections for a series of points from (minimum latitude, minimum longitude) to (maximum latitude, minimum longitude).
- Calculate min/max coordinates along south edge of output area by computing latitude/longitude to map x/y projections for a series of points from (minimum latitude, minimum longitude) to (minimum latitude, maximum longitude).
- Calculate min/max coordinates along north edge of output area by computing latitude/longitude to map x/y projections for a series of points from (maximum latitude, minimum longitude) to (maximum latitude, maximum longitude).

Note that since lines of constant latitude and/or longitude may be curved in map projection space, the extreme map x/y points may not correspond to the four PROJBOX corners.

b).2. Minbox/Maxbox Framing Determine the frame in output space for the minbox or maxbox north-up product. The actual frame is determined based on the optimal band's pixel size, but the frame is the same for every band.

b).2.1 Minbox Framing Calculate the MINBOX frame bounds using the active area corner points for each band.

1. Call projtran (see below) to get the output map projected x/y, for each active area corner point for each image band.

2. Find the minimum and maximum output proj x/y from the full set of active area corner points.

3. Pad the min and max output projection x/y to make them a multiple of pixsize.

4. Fill in the corners for the grid in the order of UL, LL, UR, LR and Y/X coords.
   UL = min x, max y
   UR = max x, max y
   LL = min x, min y
   LR = max x, min y

5. Find the number of lines and samples for the grid, for each specified band number.
   lines = (max y - min y)/pixsize + 1
   samples = (max x - min x)/pixsize + 1

b).2.2 Maxbox Framing Calculate the MAXBOX product frame bounds using the projected corners of each band/SCA.

1. Find the four image corners in input space for each SCA and band.
   UL - (1, first_pixel)
   UR - (1, last_pixel)

LL - (NLines, first_pixel)
LR - (NLines, last_pixel)

2. Call the forward model (see below) to get the output lat/long, for each corner point.

3. Call projtran (see below) to get the output map projected x/y, for each corner point.

4. Find the minimum and maximum output proj x/y from the full set of corner points.

5. Pad the min and max output projection x/y to make them a multiple of pixsize.

6. Fill in the corners for the grid in the order of UL, LL, UR, LR and Y/X coords.
   UL = min x, max y
   UR = max x, max y
   LL = min x, min y
   LR = max x, min y

7. Find the number of lines and samples for the grid, for each specified band number.
   lines = (max y - min y)/pixsize + 1
   samples = (max x - min x)/pixsize + 1

8. Call projtran to convert the map projection Y/X coordinates of the output product corners to latitude/longitude.

9. Replace the active area corner coordinates for each band with the converted output product corner coordinates.

b).2.3. Pad Corners Pad the input corners by a defined factor of the pixel size. The x/y min and max values are input for the corner locations. These values are padded by PADVAL * the pixel size. The newly padded x/y min and max values are returned, replacing the original values.

ixmin = int (Xmin/(PADVAL*pixsize))
Xmin = ixmin*PADVAL*pixsize
ixmax = int (Xmax/(PADVAL*pixsize))+1
Xmax = ixmax*PADVAL*pixsize
iymin = int (Ymin/(PADVAL*pixsize))
Ymin = iymin*PADVAL*pixsize
iymax = int (Ymax/(PADVAL*pixsize))+1
Ymax = iymax*PADVAL*pixsize

c) Path Oriented Framing

Provide a path-oriented projection that is framed to a nominal WRS scene. The projection, pixel size, and the path and row of the scene must be defined.

c).1. Calculate Center and Rotation Angle

Calculate the scene center and rotation angle for a nominal WRS scene. The WRS path and row of the input scene and the projection parameters are needed as input. The nominal WRS scene center lat/long and rotation angle for the given projection are returned. The algorithm has the following steps:

Convert input angles to radians:

Inclination_Angle_R = Pi / 180 * Inclination_Angle
Long_Path1_Row60_R = Pi / 180 * Long_Path1_Row60

Compute the Earth's angular rotation rate:
earth_spin_rate = 2 * Pi / (24 * 3600)

Note: We use the solar rotation rate rather than the sidereal rate in order to account for the orbital precession which is designed to make the orbit sun synchronous. Thus, the apparent Earth angular velocity is the inertial (sidereal) angular velocity plus the precession rate which, by design, is equal to the solar angular rate.

Compute the spacecraft's angular rotation rate:
SC_Ang_Rate = 2 * Pi * WRS_Cycle_Orbits / (WRS_Cycle_Days*24*3600)

Compute the central travel angle from the descending node:
Central_Angle = (Row - Descending_Node_Row)/Scenes_Per_Orbit*2*Pi

Compute the WRS geocentric latitude:
WRS_GCLat = asin(-sin(Central_Angle) * sin(Inclination_Angle_R))

Compute the longitude of Row 60 for this Path:
Long_Origin = Long_Path1_Row60_R - (Path-1) * 2*Pi/WRS_Cycle_Orbits

Compute the WRS longitude:
Delta_Long = atan2(tan(WRS_GCLat)/tan(Inclination_Angle_R),
cos(Central_Angle)/cos(WRS_GCLat))
WRS_Long = Long_Origin - Delta_Long - Central_Angle *
Earth_Spin_Rate / SC_Ang_Rate

Make sure the longitude is in the range +/- Pi:
While (WRS_Long > Pi)
WRS_Long = WRS_Long - 2*Pi
While (WRS_Long < -Pi)
WRS_Long = WRS_Long + 2*Pi

Compute the scene heading:
Heading_Angle = atan2(cos(Inclination_Angle_R)/cos(WRS_GCLat),
-cos(Delta_Long)*sin(Inclination_Angle_R))

Convert the WRS geocentric latitude to geodetic latitude:
WRS_Lat = atan(tan(WRS_GCLat) * (Semi_Major_Axis/Semi_Minor_Axis) *
(Semi_Major_Axis/Semi_Minor_Axis))

Convert angles to degrees:
WRS_Lat = WRS_Lat * 180 / Pi
WRS_Long = WRS_Long * 180 / Pi
Heading_Angle = Heading_Angle * 180 / Pi

Round WRS lat/long off to the nearest whole arc minute:
WRS_Lat = round(WRS_Lat*60) / 60

WRS_Long = round(WRS_Long*60) / 60

c).2. Calculate Path Oriented Frame

Calculate the center point and rotation angle, and the image corner coordinates in an SOM or UTM projection. Also calculate the first-order polynomial coefficients which map output line/sample coordinates to their corresponding output projection coordinates. Determine the frame in output space for the path-oriented product. Calculate the frame for each band. The frame must be the same for all bands.

c).2.1. Angle to Map

Convert the WRS rotation angle (from geodetic north) to a frame orientation angle in map coordinates. The following is an algorithm to compute this:

Convert the WRS scene center latitude/longitude to map projection x/y (X1, Y1) using the projtran routine.

Add 1 microradian (0.2 seconds) to the WRS scene center latitude and convert this point to map projection x/y (X2, Y2).

Compute the azimuth of this line in grid space as the arctangent of (X2-X1)/(Y2-Y1). This is the grid azimuth of geodetic north at the WRS scene center.

Add this angle to the WRS rotation angle to give the grid heading. A standard framed scene puts the satellite direction of flight at the bottom of the scene, so the scene orientation angle is the grid heading + or - 180 degrees. If the grid heading is <0 then subtract 180 degrees. If the grid heading is >0 then add 180 degrees. This is the scene orientation angle to use with the WRS scene center.

c).2.2. Path-oriented Minbox/Maxbox Frame

Calculate the path oriented frame that is large enough to contain all bands.

c).2.2.1. Calculate Path-oriented Minbox Frame

Calculate path-oriented frame for the minbox approach.

1. Compute the map projection coordinates of the four image active area corners for each band as described in step 1 of Minbox Framing.
2. Offset and rotate the scene corners to the path oriented frame using the WRS scene center map projection coordinates (X1, Y1) and orientation angle:
    a. X' = (X - X1) cos(angle) - (Y - Y1) sin(angle) + X1
    b. Y' = (X - X1) sin(angle) + (Y - Y1) cos(angle) + Y1
3. Compute the minbox frame as described in steps 2-4 of Minbox Framing.
4. Convert the rotated minbox corners back to the unrotated map projection coordinate system:
    a. X = (X' - X1) cos(angle) + (Y' - Y1) sin(angle) + X1
    b. Y = -(X' - X1) sin(angle) + (Y' - Y1) cos(angle) + Y1

c).2.2.2. Calculate Path-oriented Maxbox Frame

Calculate path-oriented frame for the maxbox approach.

1. Compute the map projection coordinates of the four image corners for the optimal band as described in steps 1-3 of Maxbox Framing.
2. Offset and rotate the scene corners to the path oriented frame using the WRS scene center map projection coordinates (X1, Y1) and orientation angle:

   a. X' = (X - X1) cos(angle) - (Y - Y1) sin(angle) + X1
   b. Y' = (X - X1) sin(angle) + (Y - Y1) cos(angle) + Y1
3. Compute the maxbox frame as described in steps 4-6 of Maxbox Framing.
4. Convert the rotated maxbox corners back to the unrotated map projection coordinate system:
   a. X = (X' - X1) cos(angle) + (Y' - Y1) sin(angle) + X1
   b. Y = -(X' - X1) sin(angle) + (Y' - Y1) cos(angle) + Y1
5. Call projtran to convert the map projection Y/X coordinates of the output product corners to latitude/longitude.
6. Replace the active area corner coordinates for each band with the converted output product corner coordinates.

d) Celestial Acquisitions

Celestial acquisitions use the same framing logic as Earth acquisitions (namely maxbox) but the output space coordinate systems are sufficiently different to merit separate discussion. For both lunar and stellar acquisitions the output space is defined in terms of directions in inertial space, defined by the ECI J2000 right ascension and declination of the OLI look vectors. In the case of stellar acquisitions, the output space "projection" uses the ECI J2000 right ascension and declination directly. For lunar acquisitions the output coordinate system is modified to use the LOS right ascension and declination offset from the lunar right ascension and declination at the time of observation. This creates a slowly rotating coordinate system that tracks the moon and is the reason for having a planetary ephemeris file as an input to this algorithm. These differences emerge in the forward model computations for celestial acquisitions where the LOS intersection logic used for Earth acquisitions is replaced by operations on the inertial lines-of-sight (after conversion to inertial right ascension and declination angles), with the resulting map projection x/y coordinates used in the Earth-view algorithms replaced by right ascension and declination (or delta-right ascension and delta-declination). Either the maxbox or minbox framing logic applied to the x/y map projection coordinates in Earth-view acquisitions is then applied to these angular celestial coordinates.

e) Lunar Acquisitions

Lunar acquisitions use either a MAXBOX or MINBOX framing type. For the case of MAXBOX the framing logic is the same as that used for Earth viewing acquisitions; determine bounding viewing angles based on all SCAs of all bands. For the case of MINBOX the minimum box, or viewing angles, are based on the SCA within which the moon resides. The bounding viewing angles for this SCA for all bands define the frame for the output image. The moon is found to reside within an SCA by using the moon's coordinates, defined by the center of acquisition time of the scene, and checking to see if these coordinates fall within a SCA. A simple point in a polygon routine is used for the check:

If coordinate to check is defined as X_m and Y_m

e1) Define rectangle using SCA corners, X_i and Y_i

e2) Find maximum Y coordinate of rectangle, Y_{max}.

e3) Define a new coordinate as:

$X_n = X_m$

Y_n = Delta + Y_{max}

Where Delta is large enough to put point (X_n, Y_n) outside of polygon

e4) Define a line from (X_m,Y_m) to (X_n,Y_n).

e5) Determine number of times the line defined in e4) intersects sides of the rectangle from e1). If the number of intersections is an odd number then the point is within the rectangle.

Stage 3 - Grid Definition

The grid definition stage determines the required size of the grid, allocates the grid structure, and computes the input space (Level 1R) line/sample locations for each grid cell.

a) Determine Number of Grid Input/Output Lines/Samples

Determine the number of input points to be stored in the grid according to the grid sampling rate or grid cell size chosen.

Loop through each band stored in the grid

Loop through each SCA stored in the grid.

Calculate the number of lines and samples stored in the grid according to the size of each grid cell and the size of the input image to be processed. Store the number of grid lines and samples calculated in the grid.

Calculate number of times grid cell size divides into Level 1R imagery

$$\text{number of grid lines} = \frac{\text{number of image lines}}{\text{grid cell size line direction}} + 1$$

$$\text{number grid samples} = \frac{\text{number of detectors per SCA}}{\text{grid cell size sample direction}} + 1$$

where:

number of image lines = number of lines in Level 1R (LOS model)
number of detectors per SCA = number of samples per SCA (LOS model)
grid cell size line direction = number of lines in one grid cell
grid cell size sample direction = number of samples in one grid cell

If the grid cell size in the line direction does not divide evenly into the number of lines in the Level 1R then increment the number of grid lines by one.

If the grid cell size in the sample direction does not divide evenly into the number of samples in the Level 1R then increment the number of grid samples by one.

b) Determine Grid Lines/Samples

Determine where each grid cell point will fall in the input Level 1R image. These grid cell points will fall at integer locations in the input imagery.

Loop through each band that is stored in the grid

Loop through each SCA stored in the grid

Initialize first grid cell line location to zero relative.

input line location grid cell$_0$ = 0

Loop until the grid cell line location is greater than or equal to the number of Level 1R lines, incrementing each new grid cell line location by the appropriate grid cell size in the line direction for the current band and SCA.

$$\text{input line location grid cell}_n = \text{input line location grid cell}_{n-1} + \text{grid cell size line direction}$$

Set last grid cell line location to the last line in Level 1R image.

$$\text{input line location grid cell}_{last} = \text{number of lines in Level 1R imagery}$$

Initialize first grid cell sample location to zero relative.

$$\text{input sample location grid cell}_0 = 0$$

Loop until the grid cell sample location is greater than or equal to the number of Level 1R samples, incrementing each new grid cell sample location by the appropriate grid cell size in the sample direction for the current band and SCA.

$$\text{input sample location grid cell}_n = \text{input sample location grid cell}_{n-1} + \text{grid cell size sample direction}$$

Set last grid cell sample location to the last sample in Level 1R image.

$$\text{input sample location grid cell}_{last} = \text{number of samples in Level 1R imagery}$$

Stage 4 - Grid Construction

Once the grid structures are created (one per SCA per band) the forward model is evaluated at every grid intersection, that is, for every Level 1R line/sample location at every elevation plane. The forward model computes the WGS84 latitude/longitude coordinates associated with each input line/sample/height point. These latitude/longitude positions are then converted to output space line/sample by projecting them to map x/y, computing the offsets (and rotation if path-oriented) from the upper-left scene corner, and scaling the offsets from meters to pixels using the pixel size.

a) Make Grid Given the number of grid lines and samples that will be sampled in the input imagery, loop on each band of each SCA, loop on number of z-planes, loop on number of input grid lines and samples calculating the corresponding output line and sample location. For each input line, sample location, and elevation, the instrument forward model function is called. This forward model function is outlined in the steps below. Additional detail on the sub-algorithms which comprise the forward model is provided in the subsection titled "Forward Model" later in this document.

The forward model uses the LOS model structure and the CPF to map an input line and sample location to an output geographic location. These are the steps that are performed whenever calculating an output geodetic latitude and longitude from an input line and sample by invoking the instrument “forward model.” The GCTP function can then be used to transform the geographic latitude and longitude to a map projection X and Y coordinate. If the output image has a “North up” orientation, then the upper left projection coordinate of the output imagery and the output pixel size can be used to transform any projection coordinate to an output line and sample location. If the map projection space is in a rotated projection space, such as having a satellite path orientation, then a transformation handling rotation is established between projection space and output pixel location. This transformation is then used in converting projection coordinates to output pixel line and sample locations.

The process listed below is performed on all bands, all elevation planes, and all SCAs present in the grid. The detector type used in the process is nominal (see the LOS Model Creation ADD for a discussion of detector types). The list explains the actions taken if a detector type other than nominal is chosen, so that it can be referenced later.

Loop on number of input grid lines.

Loop on number of input grid samples.

Read the input space (Level 1R) line/sample coordinate for this grid point.

Loop on the number of elevation planes.

Compute the height of the current elevation plane:

$$\text{height} = (z - z_{elev=0}) * \text{elevation\_increment}$$

where:
z is the index of the current z-plane and
$z_{elev=0}$ is the index of the zero elevation z-plane.

Invoke the forward model to compute the corresponding ground position latitude/longitude for this point. The general steps of the forward model are described here and are presented in more detail below.

Find Time
Find the nominal time of input sample relative to the start of the imagery. This procedure is described in the LOS Model Creation ADD and is repeated below in the Find Time sub-algorithm description.

Find LOS
Find the LOS vector for the input line/sample location using the Legendre polynomial coefficients as described below in the Find LOS sub-algorithm.

Find Attitude
Calculate the spacecraft attitude corresponding to the LOS, i.e. for the line/sample location, at the time computed above, using the Find Attitude sub-algorithm described below. Note that for Earth acquisitions the roll-pitch-yaw attitude sequence in the LOS model is relative to the orbital coordinate system whereas for celestial (lunar/stellar) acquisitions the LOS model roll-pitch-yaw sequence is with respect to the ECI J2000 coordinate system. The operations applied by the Find Attitude sub-algorithm are the same in either case.

Find Ephemeris
Calculate satellite position for line/sample using Lagrange interpolation. Reference the move_sat sub-algorithm described in the LOS Model Creation ADD and repeated below. Note that for Earth acquisitions the move_sat sub-algorithm is provided with the corrected ECEF ephemeris data from the LOS model whereas for celestial (lunar/stellar) acquisitions it will be passed the corrected ECI ephemeris.

Rotate LOS to ECEF (Earth-view) or ECI (Celestial)
Use the OLI alignment matrix in the LOS model to convert the LOS vector from sensor to ACS/body coordinates. Then apply the interpolated roll, pitch, and yaw to the LOS to convert ACS/body to orbital (Earth-view) or ECI (celestial). If Earth-view, use the ephemeris to construct the orbital to ECEF rotation matrix and use it to transform LOS to ECEF. The procedure for Earth-view scenes is described in the Attitude sub-algorithm below. For celestial acquisitions, the procedure is complete once the LOS has been rotated to ECI using the roll-pitch-yaw perturbation matrix.

Spacecraft Center of Mass to OLI Offset Correction
Adjust the spacecraft position for the offset between the spacecraft center of mass and the OLI instrument. This offset, in spacecraft body coordinates, is stored in the LOS model structure. First, convert the offset from spacecraft body frame to ECEF using the attitude perturbation matrix (body to orbital) and the orbital to ECEF matrix:

$$[\text{orbital CM to OLI}] = [\text{perturbation}][\text{body CM to OLI}]$$

$$[\text{ECEF CM to OLI}] = [\text{ORB2ECEF}][\text{orbital CM to OLI}]$$

Add the offset to the ECEF spacecraft position vector. This correction is not used for celestial (lunar/stellar) acquisitions.

Correct LOS for Velocity Aberration
The relativistic velocity aberration correction adjusts the computed LOS (ECEF for Earth-view and ECI for celestial) for the apparent deflection caused by the relative velocity of the platform (spacecraft) and target. The preparatory computations are somewhat different for Earth-view and celestial acquisitions due to the differences in target velocity.

Earth-view Case
The LOS intersection sub-algorithm described below (see Find Target Position) is invoked with an elevation of zero to find the approximate ground target position. The ground point velocity is then computed as:

$\mathbf{V}_g = \omega \times \mathbf{X}_g$

where:
$\mathbf{V}_g$ = ground point velocity
$\mathbf{X}_g$ = ground point ECEF position
ω = Earth rotation vector = $[\ 0 \quad 0 \quad \Omega_e\ ]^T$
Ω_e = Earth rotation rate in radians/second (from CPF)

The relative velocity is then:

$\mathbf{V} = \mathbf{V}_s - \mathbf{V}_g$

where Vs is the spacecraft ECEF velocity from the ephemeris data.

Correcting the Earth-View LOS
The LOS vector is adjusted based on the ratio of the relative velocity vector to the speed of light (from the CPF):

$$\boldsymbol{l}' = \frac{\boldsymbol{l} - \frac{\boldsymbol{V}}{c}}{\left| \boldsymbol{l} - \frac{\boldsymbol{V}}{c} \right|}$$

where: **l** = uncorrected LOS and **l'** = corrected LOS

Note that in this case the LOS velocity aberration correction is negative since we are correcting the apparent LOS to the true (aberration corrected) LOS. The correction is positive if we are computing the apparent LOS from the true (geometrical) LOS (see lunar case below).

Celestial (Lunar/Stellar) Case
Both lunar and stellar acquisitions use the spacecraft inertial velocity from the ephemeris data as the relative velocity. This is justified by the use of a lunar ephemeris (using the Naval Observatory's NOVAS-C package) that returns apparent places. The apparent location of the moon is already corrected for light travel time (see below) and velocity/planetary aberration due to the motion of the moon around the Earth. Thus, the residual aberration is due only to the motion of the spacecraft relative to the Earth. Thus, for both lunar and stellar acquisitions:

$\mathbf{V} = \mathbf{V}_s$

where Vs is the spacecraft ECI velocity from the ephemeris data.

Correcting the Celestial LOS
For stellar acquisitions, the LOS is corrected for aberration in the same manner as for Earth-view scenes. For lunar acquisitions, rather than correct the LOS vector, we adjust the apparent location of the moon. The lunar vector is thus adjusted based on the ratio of the relative velocity vector to the speed of light (from the CPF) as:

$$\mathbf{l}' = \frac{\mathbf{l} + \frac{\mathbf{V}}{c}}{\left| \mathbf{l} + \frac{\mathbf{V}}{c} \right|}$$

where: **l** = uncorrected LOS and **l'** = corrected LOS

The correction is positive in this case since we are computing an apparent location rather than correcting one.

LOS Intersection
For Earth-view acquisitions, intersect the LOS in ECEF with the Earth model as described in the Find Target Position sub-algorithm below. This yields the geodetic latitude, longitude, and height of the ground point.

For celestial acquisitions, convert the ECI LOS to right ascension (RA) and declination (δ) angles:

$$RA = \tan^{-1}(\frac{y}{x})$$

$$\delta = \tan^{-1}(\frac{z}{\sqrt{x^2 + y^2}})$$

where the ECI los vector is [x y z]^T.

Correct Ground Position for Light Travel Time

Since the light departing the ground point takes a finite time to arrive at the OLI sensor, there is a slight discrepancy in the corresponding time at the ground point and at the spacecraft. Since the LOS intersection logic assumed that these times were the same, a small correction can be made to correct for this light travel time delay.

Given the ECEF positions of the ground point and the spacecraft, compute the light travel time correction as follows:

Compute the distance from the ground point to the spacecraft:

$$d = \left|\mathbf{X_s} - \mathbf{X_g}\right|$$

where:

$\mathbf{X}_s$ is the spacecraft ECEF position and
$\mathbf{X}_g$ is the ground point ECEF position.

Compute the light travel time using the speed of light (from CPF):

$$ltt = \frac{d}{c}$$

Compute the Earth rotation during light travel:

$\theta = ltt * \Omega_e$ where Ω_e is the Earth angular velocity from the CPF.

Apply the light travel time Earth rotation:

$$\mathbf{X}'_g = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{X}_g$$

where:

$\mathbf{X}_g$**'** is the corrected ECEF position
X_g is the uncorrected ECEF position

Convert the corrected ECEF position to geodetic latitude, longitude and height.

Note that the light travel time correction for lunar observations due to the difference between the Earth-moon distance and the spacecraft-moon distance is neglected. This is justified by the fact that that the lunar angular rate is less than 3 microradians per second and the maximum LTT difference is about 25 milliseconds making the magnitude of this effect less than 0.1 microradians.

Convert Position to Output Space Line/Sample

The angular geodetic (latitude/longitude) or celestial (RA/declination) coordinates must be converted to the corresponding output space line/sample coordinate to complete the input space to output space mapping.

For Earth-view acquisitions this is accomplished as follows:

Calculate the map projection X/Y for the geodetic latitude and longitude.

Convert map X/Y coordinate to output line/sample location:

If the output map projection is of a path-oriented projection then the X/Y coordinate is transformed to output space with a bilinear transformation.

$$line = a_0 + a_1 * X + a_2 * Y + a_3 * X * Y$$
$$sample = b_0 + b_1 * X + b_2 * Y + b_3 * X * Y$$

where:
a_i = polynomial coefficients that map X/Y to an output line location
b_i = polynomial coefficients that map X/Y to an output sample location
X,Y = map projection coordinates

The polynomial transformation is set up to handle the rotation involved in rotating a “Map North” projection to Satellite of “Path” projection (i.e. one that has the output line coordinate system more closely aligned with the along flight path of the satellite).

If the output map projection is not path-oriented, but “North up,” the relationship between X/Y and output line/sample does not involve any rotation and the following equation is used:

$$\text{line} = \frac{\text{upper left Y} - \text{Y}}{\text{pixel size Y}}$$
$$\text{sample} = \frac{\text{X} - \text{upper left X}}{\text{pixel size X}}$$

where:
upper left Y = upper left Y projection coordinate of output image
upper left X = upper left X projection coordinate of output image
pixel size Y = output pixel size in Y coordinates
pixel size X = output pixel size in X coordinates

Note that these line and sample pixel coordinates are (0,0) relative (i.e., the center of the upper left pixel is at line,sample 0,0).

For lunar acquisitions, the right ascension and declination angles derived from the inertial LOS are offset from the nominal lunar inertial position to establish an output frame that "tracks" the apparent location of the moon. This is done as follows:

a) Compute the apparent ECI J2000 position of the moon.

1. Use the input JPL lunar ephemeris data in the NOVAS-C package to compute the ECI true-of-date (ECITOD) apparent location of the moon at the time corresponding to the current LOS (lxx_moonpos). This apparent location is provided as an ECITOD vector (i.e., including both direction and distance).
2. Apply the nutation and precession corrections (see Ancillary Data Preprocessing ADD for additional information) to convert the ECITOD vector to ECI J2000.
3. Subtract the current spacecraft ECI J2000 position vector from the lunar ECI J2000 vector to compute the spacecraft-lunar vector.
4. Compute the apparent (parallax corrected) right ascension, declination, and spacecraft-lunar distance from the spacecraft-lunar vector (by invoking exx_cart2sph.

b) Compute the differences between the LOS right ascension and declination and the apparent lunar right ascension and declination.

c) Normalize the nominal angular pixel size by the ratio of the current spacecraft-moon distance (computed above) and the nominal spacecraft-moon distance. The nominal distance is computed at the acquisition center time.

$$psize_{current} = psize_{nominal} * distance_{nominal} / distance_{current}$$

d) Divide the angular distances computed in b) above by the normalized pixel size computed in c) above. This yields the moon-relative line/sample coordinate. This is the coordinate space in which lunar images are framed, so the offset between these coordinates and the lunar scene upper left corner coordinates yields the output space line/sample for the current grid point.

For stellar acquisitions, the right ascension and declination angles derived from the inertial LOS are used directly. The offsets relative to the scene upper left corner (in right ascension/declination space) are computed and divided by the angular pixel size to compute output space line/sample coordinates.

One additional note regarding the celestial acquisition scene framing is in order. Since right ascension, like longitude, increases eastward, and declination, like latitude, increases northward, and given that celestial images are looking "up" rather than "down", the right ascension-x, declination-y coordinate system is left-handed. This can lead to the moon being apparently inverted left-to-right in the output image. This is not important for the applications (e.g., band registration characterization) in which the lunar images are to be used. If "anatomically correct" lunar images are required, some changes to the framing logic may be necessary.

The line and sample location calculated is stored in the grid structure. This line/sample location is then the output location for the corresponding input line/sample and the current elevation (current grid line/sample input locations).

b) Calculate Jitter Sensitivity Coefficients The forward model is invoked multiple times at each grid intersection to compute the effect that small attitude perturbations about each spacecraft axis have on the input space to output space line/sample mapping. This is done at each grid point as follows:

Save the current grid point input line/sample as in_line/in_samp and the current grid point output line/sample as line0/samp0.

For each spacecraft axis (roll-pitch-yaw) :

1. Perturb the attitude about the selected axis by 1 microradian.
2. Use the forward model to compute the output line/sample corresponding to the current input line/sample using the perturbed attitude and store the result in line[0]/samp[0].
3. Perturb the input line number by 1 line (delta_line = 1) and recompute the corresponding output line/sample, storing the result in line[1]/samp[1].
4. Restore the input line number to in_line and perturb the input sample number by 1 sample (delta_samp = 1) and recompute the corresponding output line/sample, storing the result in line[2]/samp[2].
5. Calculate the output space to input space line/sample sensitivities as:
   a. delta_oline_per_iline = (line[1]–line[0]) / delta_line
   b. delta_oline_per_isamp = (line[2]–line[0]) / delta_samp
   c. delta_osamp_per_iline = (samp[1]–samp[0]) / delta_line
   d. delta_osamp_per_isamp = (samp[2]–samp[0]) / delta_samp
6. Invert the resulting 2-by-2 sensitivity matrix to find the input line/samp per output line/samp sensitivities:
   a. determinant = delta_oline_per_iline * delta_osamp_per_isamp – delta_oline_per_isamp * delta_osamp_per_iline
   b. delta_iline_per_oline = delta_osamp_per_isamp / determinant
   c. delta_iline_per_osamp = -delta_oline_per_isamp / determinant
   d. delta_isamp_per_oline = -delta_osamp_per_iline / determinant
   e. delta_isamp_per_osamp = delta_oline_per_iline / determinant
7. Apply the input line/samp per output line/samp sensitivities to the output line/samp offset due to the attitude perturbation, to find the equivalent input space offset :
   a. d_iline = delta_iline_per_oline * (line[0] – line0) + delta_iline_per_osamp * (samp[0] – samp0)
   b. d_isamp = delta_isamp_per_oline * (line[0] – line0) + delta_isamp_per_osamp * (samp[0] – samp0)
8. Divide by the attitude perturbation to compute the input line/sample to attitude jitter sensitivities for this axis at this grid point:
   a. line_sens[axis] = -d_iline / perturbation
   b. samp_sens[axis] = -d_isamp / perturbation

   Where :

   line_sens[] is the array of roll-pitch-yaw line sensitivities for the grid.
   samp_sens[] is the array of roll-pitch-yaw sample sensitivities for the grid.
   perturbation is the 1 microradian attitude perturbation introduced in step 1.

   Note that the sign of the sensitivities is inverted in this calculation. This is done because the sensitivities will be used to compute the equivalent input space corrections needed to compensate for an attitude disturbance. So, since d_iline is the input space line offset that is equivalent to one microradian of jitter for the current axis, an offset of –d_iline will compensate for this jitter.

A 2-by-3 array containing the line and sample sensitivity coefficients for the roll, pitch, and yaw axes is stored for each grid point.

c) Calculate Map Coefficients Bilinear mapping coefficients for each grid cell are calculated for mapping from input location to output location (forward mapping) and for mapping from output location to input location (inverse mapping). A separate mapping function is used for lines and

samples. This equates to four mapping functions. A set of four mapping functions is calculated for each grid cell, for each SCA, for every band, and for every elevation plane that is stored in the grid.

The following methodology is used for calculating one set of four bilinear mapping equations:

A 9x4 matrix is used to fit nine points within a grid cell. The matrix equation takes the form of:

$$[A][coeff] = [b]$$

In this equation, matrix A is 9x4, vector b is 9x1, and the coefficient matrix is 4x1. The coefficient matrix, [*coeff*], can be solved to obtain the mapping coefficients as:

$$[coeff] = \left[A^T A\right]^{-1}\left[A^T b\right]$$

In the case of solving for an equation to map an input line and sample location to an output sample location, belonging to one grid cell, the matrices can be defined as:

$A_{n,0} = 1$ where n=0,8
$A_{0,1}$ = upper left input sample location for current grid cell
$A_{1,1}$ = upper right input sample location for current grid cell
$A_{2,1}$ = lower left input sample location for current grid cell
$A_{3,1}$ = lower right input sample location for current grid cell
$A_{4,1} = (A_{0,1}+A_{1,1}+A_{2,1}+A_{3,1})/4$
$A_{5,1} = (A_{0,1}+A_{1,1})/2$
$A_{6,1} = (A_{1,1}+A_{3,1})/2$
$A_{7,1} = (A_{2,1}+A_{3,1})/2$
$A_{8,1} = (A_{2,1}+A_{0,1})/2$
$A_{0,2}$ = upper left input line location for current grid cell
$A_{1,2}$ = upper right input line location for current grid cell
$A_{2,2}$ = lower left input line location for current grid cell
$A_{3,2}$ = lower right input line location for current grid cell
$A_{4,2} = (A_{0,2}+A_{1,2}+A_{2,2}+A_{3,2})/4$
$A_{5,2} = (A_{0,2}+A_{1,2})/2$
$A_{6,2} = (A_{1,2}+A_{3,2})/2$
$A_{7,2} = (A_{2,2}+A_{3,2})/2$
$A_{8,2} = (A_{2,2}+A_{0,2})/2$
$A_{n,3} = A_{n,1}*A_{n,2}$ where n=0…8

b_0 = upper left output sample location for current grid cell
b_1 = upper right output sample location for current grid cell
b_2 = lower left output sample location for current grid cell
b_3 = lower right output sample location for current grid cell
$b_4 = (b_0+b_1+b_2+b_3)/4$
$b_5 = (b_0+b_1)/2$
$b_6 = (b_1+b_3)/2$
$b_7 = (b_2+b_3)/2$
$b_8 = (b_2+b_0)/2$

The line and sample locations listed above are defined at the grid cell corners coordinates. The points interpolated in between the grid cell line segments provide stability for what could be, most notably a mapping that involves a 45^o rotation, an ill-defined solution if only four points were used in the calculation. The set of coefficients define a bilinear mapping equation of the form:

$$sample_o = coeff_0 + coeff_1 * sample_i + coeff_2 * line_i + coeff_3 * sample_i * line_i$$

where:

$sample_o$ = output sample location
$sample_i$ = input sample location
$line_i$ = input line location

The forward mapping equations, mapping input line and sample locations to output line locations can be solved by swapping output line locations for output sample locations in the matrix [b]. The reverse mapping equations, mapping output locations to input line and sample, can be found by using output line and sample locations in the [A] matrix and the corresponding input sample and then line locations in the [b] matrix.

c).1. Calculate Forward Mappings

Using the Calculate Map Coefficients algorithm described above generate the mapping polynomial coefficients needed to convert from a line/sample in input space (satellite) to one in output space (projection). Coefficients for every cell in the grid are generated.

c).2. Calculate Inverse Mappings

Using the Calculate Map Coefficients algorithm described above generate the mapping polynomial coefficients needed to convert from a line/sample in output space (projection) to one in input space (satellite). Coefficients for every cell in the grid are generated.

Stage 5 - Finalize the Grid

The final stage of grid processing generates the global (rough) mapping coefficients, used to initially identify the appropriate grid cell, and computes the parallax sensitivity coefficients, used to correct for even/odd detector offset effects, for each grid cell.

a) Calculate Rough Mapping Coefficients

Calculate the rough mapping coefficients for the grid. The rough polynomial is a set of polynomials used to map output line and sample locations to input line and sample locations. The rough polynomial is generated using a large number of points distributed over the entire scene, and by calculating a polynomial equation that maps an output location to an input location. The rough polynomial is only meant to get a “close” approximation to the input line and sample location for a corresponding output line and sample location. Once this approximation is made, the value can be refined to get a more accurate solution. A rough mapping polynomial is found for every SCA, for every band, and for every elevation plane that is stored in the grid file.

Bilinear mapping was found to be sufficient for the rough mapping. The mapping function therefore looks like the ones used for each individual grid cell. However, the set up of the matrices to solve for the mapping coefficients is different:

$$\underset{Nx4}{[A]}\underset{4x1}{[coeff]} = \underset{Nx1}{[b]}$$

Where the matrix [A] is defined by the output line and sample locations, matrix [b] is defined by either the input lines or input samples, and N is equal to the total number of points stored in the grid for one elevation plane, of one band, for a single SCA. The rough polynomial is therefore found by using all the point locations stored in the grid for a given band and elevation plane for a single SCA. There is one mapping for output line and sample location to input sample location and one mapping for output line and sample location to input line location.

Grid Cell Polynomial
Calculate a "rough" mapping of output to input lines/samples. These coefficients are used as a first order approximation to an inverse line-of-sight model. This polynomial is used to initially locate the grid cell to be used in the resampling process, providing a starting point for the more accurate inverse model based on individual grid cell parameters.

b) Calculate Detector Offsets Computes the detector offset values and stores linear mapping coefficients associated with detector offsets in the grid structure. Using the zero elevation plane, for each band and each SCA, loop on the input lines and samples calculating the odd/even detector offsets. The detector offsets are set up to account for the geometric differences between the odd/even, secondary, and tertiary detectors and the “nominal” set of detectors. (See the LOS Model Creation ADD). These differences are considered to be consistent between actual and nominal detectors when they occur under the same acquisition conditions, i.e. they are slowly varying. These actual to nominal detector differences are due to the imperfect trade-off between space (detector offset) and time (detector delay) that is made when we temporally shift (through the use of Level 1R image fill) the even/odd and deselected detectors to compensate for their spatial offsets on the focal plane. The degree to which this time/space trade is imperfect varies with height and, so, the corrections derived here and stored in the grid structure, are functions of detector offset and height.

There are also the sub-pixel detector specific offsets that are stored in the CPF. These "exact" detector specific offsets are accounted for in the resampling process. Note that the potential for deselected detectors has made it necessary to also store per-detector full-pixel offsets in the CPF (and LOS model). As a result, this detector offset sensitivity logic has been changed to compute the offset sensitivity per pixel of detector offset rather than a fixed value derived from the static even/odd detector offset. The routine ols2ils listed below, used for mapping an output line and sample to an input line and sample using the geometric grid, is discussed in the Image Resampling ADD.

Loop on number of bands stored in grid
    Loop on number of SCAs stored in grid
        Loop on lines and samples stored in the grid

            Get the maximum detector offset value for this band from the CPF.

            Calculate the output line/sample location for the current input line and sample and the zero elevation plane, calculated using the forward model (see below) with the detector location set to MAXIMUM. This detector type is the same as ACTUAL but uses the maximum detector offset rather than the detector-specific value.

Map calculated output line/sample back to input space using the geometric grid and ols2ils.

Delta line/sample per pixel of offset are calculated by:

$\Delta line_0$ = (nominal line - mapped line) / max offset
$\Delta sample_0$ = (nominal sample - mapped sample) / max offset

where:
nominal line = current grid cell line location
mapped line = input line location from ols2ils mapped "maximum" output line
nominal sample = current grid cell sample location
mapped sample = input sample location from ols2ils mapped "maximum" output sample
max offset = detector offset used in the MAXIMUM forward model calculations

These delta lines and samples represent the input space correction necessary to compensate for the difference between nominal and actual detectors per pixel of detector offset, for the zero elevation plane.

Repeat these calculations for the maximum elevation plane to compute $\Delta line_H$ and $\Delta sample_H$ where H is the elevation corresponding to the maximum z-plane.

Compute the line and sample even/odd offset sensitivity coefficients:

$c_0 = \Delta line_0$
$c_1 = (\Delta line_H - \Delta line_0) / H$
$d_0 = \Delta sample_0$
$d_1 = (\Delta sample_H - \Delta sample_0) / H$

Note that c_0 and d_0 are in units of pixels per pixel and c_1 and d_1 are in units of pixels per meter per pixel.

These c_i and d_i coefficients are stored in the projection grid to be used during the resampling process.

Output Line/Sample to Input Line/Sample

Map output space line/sample locations back into its corresponding input space line/sample locations. This is done using the "rough" polynomial from the grid to determine an initial guess at an input space line and sample. From this initial guess a grid cell row and column is calculated and the inverse coefficients for that cell are retrieved from the grid. These coefficients are used to determine an exact input space line and sample (in extended space).

Find Grid Cell

This utility function finds the correct cell that contains the output line/sample. It finds the correct grid cell containing the output pixel by first determining the set of grid cells to be checked. It then calls a

routine to perform a "point in polygon" test on each of these grid cells to determine if the pixel does indeed fall within that grid cell.

Forward Model
Having described the grid generation procedure we now turn to the forward model, referred to extensively above, in more detail.

For a given line, sample and band, propagate the forward model to determine a latitude and longitude for the specified point. This involves finding the time of the observation, constructing the instrument line-of-sight, calculating the spacecraft attitude and ephemeris for the observation time, and intersecting the projected line-of-sight with the Earth's surface. The entire forward model procedure is referred to as LOS projection and is described step by step below.

a) Project LOS
Find the position where the line of sight vector intersects the Earth's surface. It invokes the following sub-algorithms:

a).1. Find Time Find the time into the scene given the line, sample, and band. The input sample number is 0-relative and relative to the SCA. The accounting for the odd/even, secondary, and tertiary detector offsets is based on the value of the dettype variable which may be NOMINAL, ACTUAL, MAXIMUM or EXACT. Note that the EXACT selection is treated the same as ACTUAL. This is due to the fact that even though fractional-pixel detector offsets can occur, the compensating time shifts implemented by inserting fill pixels can only be introduced in whole-line increments. So, the sub-pixel difference between the ACTUAL and EXACT detector types affects only the LOS angle not the time. The MAXIMUM detector type represents a theoretical offset that is used for calculate the odd/even offset, or parallax, coefficients within the grid. This maximum is stored as #define in the prototype code, called MAX_DET_DELAY.

Due to the staggered odd/even and multiple pixel select detectors, a nominal and an actual time can be found in a scene. If the current position within the image is given as a line and sample location, the two different "types" of times for multispectral pixels are calculated by:

```
if detector type is set to MAXIMUM
    detector_shift_x = maximum_detector_shift
    l0r_fill_pixels = round(detector_shift_x) + nominal_fill
else
    detector_shift_x = shift stored in geometric model
     l0r_fill_pixels = Fill from L0rp (also stored in geometric model)

time_index = MS_line - l0r_fill_pixels
if ( time_index < 0 ) time_index = 0
if (time_index > (num_time_stamps - 1)) time_index = num_time_stamps - 1

MS_actual_time = line_time_stamp[time_index] - MS_settle_time  - MS_integration_time/2
                    + (MS_line - l0r_fill_pixels - time_index) * MS_sample_time

MS_nominal_time = MS_actual_time + (l0r_fill_pixels – nominal_fill) * MS_sample_time
```

where:

- MS_line is the zero-referenced multispectral line number (N).
- l0r_fill_pixels is the total amount of even/odd detector alignment fill to be inserted at the beginning of the pixel column associated with the current detector. This table is stored in the LOS model.
- num_time_stamps is the total number of time codes (data frames) in the image. It is tested to ensure that time_index, the line_time_stamp index, does not go out of bounds.
- detector_shift_x (unless type is MAXIMUM) is the amount of even/odd detector offset for the current detector from the LOS model detector delay table. It is rounded to the nearest integer pixel because time offsets can only occur in whole line increments. This detector shift is stored within the geometric model.
- MS_settle_time is a small sample and hold time delay constant.
- nominal_fill is the nominal fill associated with current band and SCA.
- maximum_detector_shift is the theoretical offset used in calculating the geometric effects associated with the odd/even offset of the detectors.

The MS_settle_time correction is expected to be a small (tens of microseconds) constant offset that should be captured in the CPF. The detector_shift_x offset parameter from the LOS model detector delay table is rounded to include the effects of even/odd detector stagger and detector deselect but not the detector-specific sub-pixel offsets.

For the panchromatic band the corresponding equations for a pan detector in the two pan lines (2N and 2N+1) associated with MS line N are computed as:

```
if detector type is set to MAXIMUM
    detector_shift_x = maximum_detector_shift
    l0r_fill_pixels = round(detector_shift_x) + nominal_fill
 else
    detector_shift_x = shift stored in geometric model
     l0r_fill_pixels = Fill from L0rp (also stored in geometric model)

time_index = floor( (pan_line - l0r_fill_pixels)/2 )
if ( time_index < 0 ) time_index = 0
if (time_index > (num_time_stamps - 1)) time_index = num_time_stamps - 1

Pan_actual_time = line_time_stamp[time_index] - Pan_settle_time - Pan_integration_time/2
                  + (pan_line - l0r_fill_pixels - 2*time_index)*Pan_sample_time

Pan_nominal_time = Pan_actual_time + (l0r_fill_pixels – nominal_fill) * Pan_sample_time
```

where:

- pan_line is the zero-referenced panchromatic line number (2N or 2N+1).
- l0r_fill_pixels is the total amount of even/odd detector alignment fill to be inserted at the beginning of the pixel column associated with the current detector. These values are stored in the LOS model. Note that these values will always be even for the panchromatic band.
- num_time_stamps is the total number of time codes (data frames) in the image. It is tested to ensure that time_index, the line_time_stamp index, does not go out of bounds.
- detector_shift_x (unless type is MAXIMUM) is the amount of even/odd detector offset for the current detector from the LOS model detector delay table. It is rounded to the nearest

integer pixel because time offsets can only occur in whole line increments. This detector shift is stored within the geometric model.

- Pan_settle_time is a small sample and hold time delay constant.
- nominal_fill is the nominal fill associated with current band and SCA.
- maximum_detector_shift is the theoretical offset used in calculating the geometric effects associated with the odd/even offset of the detectors.

For the panchromatic band, the l0r_fill_pixels and detector_shift_x parameters are in units of panchromatic pixels.

a).2. Find LOS Find the line of sight vector in sensor coordinates, using the Legendre polynomial LOS model stored in the LOS model, as follows:

Find normalized detector for Legendre polynomial:

$$\text{normalized detector} = \frac{2*(\text{current detector})}{(\text{number of detectors} - 1)} - 1$$

where:

current detector = sample location (in the range 0 to number of detectors-1)
number of detectors = number of detectors (samples) for current band and SCA (from LOS model)

Find across track (y) and along track (x) angles:

$$x = coef\_x_0 + coef\_x_1 * (\text{normalized detector}) + coef\_x_2 * (1.5 * (\text{normalized detector})^2 - 0.5)$$

$$y = coef\_y_0 + coef\_y_1 * (\text{normalized detector}) + coef\_y_2 * (1.5 * (\text{normalized detector})^2 - 0.5)$$

where:

coef_x = Legendre coefficients for along track direction
coef_y = Legendre coefficients for across track direction

(Note: coef_x and coef_y are read from the CPF and stored in the LOS model)

If LOS requested is ACTUAL, add the whole pixel detector shift (detector, band, and SCA dependent for OLI) from the LOS model. This detector shift is only in the along track direction. Note that the LOS model contains the combined whole pixel and sub-pixel detector offset, so it must be rounded to the integer part for the ACTUAL detector type and left unrounded for the EXACT detector type.

x = x + round(detector_shift_x) * IFOV

If LOS requested is EXACT, then add individual detector offsets (detector number, band, and SCA dependent). This detector shift is in both the along and across track directions. These values are stored within the LOS model.

x = x + (detector_shift_x) * IFOV
y = y + (detector_shift_y) * IFOV

Note that the detector_shift_y parameter, from the LOS model detector delay table, is always sub-pixel. See LOS Model Creation ADD for further explanation of NOMINAL/ACTUAL/EXACT line of sight.

If the LOS request in MAXIMUM then add the maximum, or theoretical, detector offset.

x = x + (maximum_detector_shift_x) * IFOV

Calculate LOS vector.

$$[\text{los}] = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Normalize LOS.

$$\vec{\text{los}} = \frac{\vec{\text{los}}}{\|\vec{\text{los}}\|}$$

a).3. Find Attitude

Find the precise roll, pitch and yaw at the specified time. This routine uses the "corrected" version of the attitude data stored in the OLI LOS model. This attitude data sequence includes the effects of ground control point precision correction (if any).

Find the current time relative to attitude data start time stored in the LOS model.

dtime = time + image epoch time – attitude epoch time

Note:

time = nominal time of input sample relative to the start of the image epoch time = image start time from LOS model, only need seconds of day field since all epochs are adjusted to the same day.

attitude epoch time = attitude data start time from LOS model, only need seconds of day field since all epochs are adjusted to the same day.

Find index into attitude data (stored in model) corresponding to dtime:

$$\text{index} = \text{floor}\left(\frac{\text{dtime}}{\text{attitude sampling rate}}\right)$$

where:

attitude sampling rate = sample period from LOS model

This attitude index determination could also be implemented as a search through the attitude data time stamps which are stored in the LOS model. The selected index would be the index of the last time that does not exceed dtime.

Attitude is found by linearly interpolating between the attitude values located at index and index+1 using the corrected attitude sequence from the LOS model:

$$w = \frac{\text{fmod}(\text{dtime}, \text{attitudesampling rate})}{\text{attitudesampling rate}}$$

$$\text{roll} = \text{model roll}_{index} + (\text{model roll}_{index+1} - \text{model roll}_{index}) * w$$
$$\text{pitch} = \text{model pitch}_{index} + (\text{model pitch}_{index+1} - \text{model pitch}_{index}) * w$$
$$\text{yaw} = \text{model yaw}_{index} + (\text{model yaw}_{index+1} - \text{model yaw}_{index}) * w$$

a).3.i. Find Jitter Find the high frequency roll, pitch and yaw corrections at the specified input image line/sample coordinate. This routine uses the jitter table stored in the OLI LOS model. This table is time aligned with the OLI panchromatic band line sample times, so the jitter table look-up proceeds directly from the input line/sample coordinates:

Find the current detector number from the input sample location:
detector = round(sample)
Verify that the detector is in the valid range for this band (return error if not).
Look up the number of L0R fill pixels for this detector (from the fill table).
Calculate the jitter table index:
If (band = pan)
Index = round(line) – l0r_fill_pixels
Else
Index = 2*(round(line) – l0r_fill_pixels)
Verify that jitter table index is within the valid range for the table (return zeros if not).
Extract the roll-pitch-yaw jitter values for the current index from the jitter table and return these values.

Note that the jitter values are a direct look-up without interpolation. This does not compromise accuracy because this function is only used for cases of EXACT detector projection (e.g., the OLI data simulator) for which the input line/sample coordinates are integers. The jitter values extracted by Find Jitter are added to the low frequency roll-pitch-yaw values interpolated by Find Attitude by the calling procedure Get LOS when the EXACT option is in force.

a).4. Move Satellite Sub-Algorithm Compute the satellite position and velocity at a delta time from the ephemeris reference time using Lagrange interpolation. This is a utility sub-algorithm that accesses the "corrected" version of the model ephemeris data to provide the OLI position and velocity at any specified time. Since the model ephemeris arrays are inputs to this sub-algorithm it will work with either the ECI or ECEF ephemeris data.

Calculate time of current line/sample relative to start time of ephemeris start time.

reference time = time + image epoch time – ephemeris epoch time

where:
time = nominal time of input sample relative to the start of the imagery
image epoch time = image start time from LOS model, only need seconds of day since all epochs are on same day.
ephemeris epoch time = ephemeris start time from LOS model, only need seconds of day since all epochs are on same day.

Find index into ephemeris data stored in model.

$$\text{index} = \text{floor}\left(\frac{\text{reference time}}{\text{ephemeris time steps}} - \frac{\text{number of Lagrange points}}{2} \right)$$

where:

ephemeris time steps = time between ephemeris samples
number of Lagrange points = number of points to use in Lagrange interpolation

Use Lagrange interpolation to calculate satellite position and velocity in ECEF (or ECI, depending on which sequence is provided) coordinates at time of current line/sample.

X = Lagrange(model satellite ECEF/ECI x[index])
Y = Lagrange(model satellite ECEF/ECI y[index])
Z = Lagrange(model satellite ECEF/ECI z[index])
XV = Lagrange(model satellite ECEF/ECI vx[index])
YV = Lagrange(model satellite ECEF/ECI vx[index])
ZV = Lagrange(model satellite ECEF/ECI vx[index])

where:

X = satellite x coordinate
Y = satellite y coordinate
Z = satellite z coordinate
XV = satellite x velocity
YV = satellite y velocity
ZV = satellite z velocity

a).5. Convert Sensor LOS to Geocentric

Find the line of sight vector from the spacecraft to a point on the ground by transforming the line of sight vector in sensor coordinates to perturbed spacecraft coordinates.

Use the OLI alignment matrix in the LOS model to convert the LOS vector from sensor to body. Then apply roll, pitch, and yaw to the LOS to convert body to orbital. Finally, use the ephemeris to construct the orbital to ECEF rotation matrix and use it to transform LOS to ECEF.

First, using the 3x3 ACS to instrument alignment transformation matrix stored in the LOS model, calculate the instrument to ACS transformation matrix.

$$[\text{Instrument to ACS}] = [\text{ACS to Instrument}]^{-1}$$

Transform LOS from Instrument to ACS/body coordinates.

$$[\text{navigation los}] = [\text{Instrument to ACS}][\text{los}]$$

Calculate attitude perturbation matrix using interpolated attitude values. Note that these values include the effects of precision LOS correction (if any) as these will be built into the "corrected" attitude stream in the LOS model. The Earth-view acquisitions the roll-pitch-yaw values will be with respect to the orbital coordinate system but for celestial acquisitions they will be with respect to ECI.

Calculate perturbation matrix, [perturbation], due to roll, pitch, and yaw:

$$[\text{attitude perturbation}] = [Y_{yaw}][P_{pitch}][R_{roll}] =$$

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

Calculate new LOS in orbital coordinates (Earth-view) or ECI (celestial) due to attitude perturbation:

$$[\text{perturbation los}] = [\text{perturbation}][\text{navigation los}]$$

For Earth-view acquisitions, calculate the transformation from Orbital Coordinates to ECEF. The position and velocity vectors used in calculating the transformation are those calculated above. These vectors are in ECEF allowing the LOS to be transformed from the instrument coordinate system to the ECEF coordinate system.

Transform perturbed LOS from Orbital to ECEF.

$$[\text{ECEF los}] = [\text{ORB2ECEF}][\text{perturbation los}]$$

For celestial acquisitions, the ECI los ([perturbation los]) is returned.

a).6. Find Target Position

Finds the position where the line of sight vector intersects the Earth's surface.

Intersect the LOS in ECEF with the Earth model calculating the target ECEF vector. The ECEF vector is then used to compute the geodetic latitude and the longitude of the intersection point.

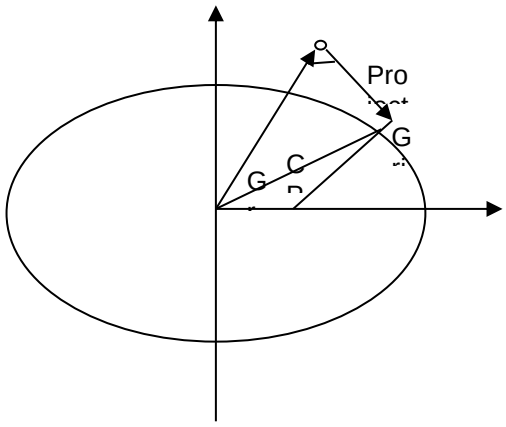


Figure 7: Intersecting LOS with Earth model

Where:

rs = satellite position vector
re = geocentric Earth vector
los = line-of-sight vector

Intersect LOS with ellipsoid

a) Rescale vectors with ellipsoid parameters.

$$rs' = \left[\frac{rsx}{a} \quad \frac{rsy}{a} \quad \frac{rsz}{b} \right]$$
$$re' = \left[\frac{rex}{a} \quad \frac{rey}{a} \quad \frac{rez}{b} \right]$$
$$los' = \left[\frac{losx}{a} \quad \frac{losy}{a} \quad \frac{losz}{b} \right]$$

where:
a = semi-major axis of Earth ellipsoid
b = semi-minor axis of Earth ellipsoid
rs' = rescaled satellite position vector
re' = rescaled geocentric Earth vector
los' = rescaled LOS vector

b) From the Law of Cosines

$$|re'|^2 = |d * los'|^2 + |rs'|^2 - 2|d * los'||rs'|\cos(\delta)$$

where:
d = los' vector length
δ = angle between rs' and los'

$$\cos(\delta) = \frac{los' \bullet rs'}{|los'||rs'|}$$

By definition | re' | = 1

Rearranging the equation determined from the Law of Cosines in terms of the constant d.

$$d^2|los'|^2 + 2d(los' \bullet rs') + |rs'|^2 - 1$$

Solving for d using the quadratic equation.

$$d = \frac{-\left|los' \bullet rs'\right| - \sqrt{\left|los' \bullet rs'\right|^2 - \left|los'\right|^2 \left(\left|rs'\right|^2 - 1\right)}}{\left|rs'\right|^2}$$

c) Compute new target vector.

$$re' = rs' + d * los'$$

d) Rescale target vector.

$$re = \left[a * rex' \quad a * rey' \quad b * rez'\right]$$

e) Compute Geodetic coordinates (see Geocentric to Geodetic below).

$$(rex, rey, rez) \Rightarrow (\varphi_0, \lambda_0, h_0)$$

If target height (H), or elevation corresponding to current z plane, is not zero:

Initialize:
Target vector: rt=re
Target height: h_0=0

Iterate until Δh =(h_i-H) is less than TOL

a) Calculate delta height.

$\Delta h = h_i - H$

b) Compute length of LOS.

$$d = \sqrt{(rtx - rsx)^2 + (rty - rsy)^2 + (rtz - rsz)^2}$$

where:
d = length of LOS vector
rt = target vector
rs = spacecraft position vector

c) Compute LOS /height sensitivity.

$$q = n \bullet los$$

Where **n** is a vector normal to the ellipsoid surface.

$$n = \left[\cos(\varphi_i)\cos(\lambda_i) \quad \cos(\varphi_i)\sin(\lambda_i) \quad \sin(\lambda_i)\right]^T$$

and:
q = LOS height sensitivity coefficient

los = LOS unit vector
ϕ_i = current estimate of ground point latitude
λ_i = current estimate of ground point longitude

d) Adjust LOS.

$$d = d + q * \Delta h$$

e) Re-compute target vector.

$$rt = rs + d * los$$

f) Calculate new geodetic coordinates and corresponding height above ellipsoid.

$$(rtx, rty, rtz) \Rightarrow (\varphi_{i+1}, \lambda_{i+1}, h_{i+1})$$

Calculate the geodetic latitude and longitude from the final ECEF vector.

a).7. Geocentric to Geodetic The relationship between ECEF and geodetic coordinates can be expressed simply in its direct form:

$e^2 = 1 - b^2 / a^2$
$N = a / (1 - e^2 \sin^2(\varphi))^{1/2}$
$X = (N + h) \cos(\varphi) \cos(\lambda)$
$Y = (N + h) \cos(\varphi) \sin(\lambda)$
$Z = (N (1-e^2) + h) \sin(\varphi)$

where:

X, Y, Z	-	ECEF coordinates
φ, λ, h	-	Geodetic coordinates (lat ϕ, long λ, height h)
N	-	Ellipsoid radius of curvature in the prime vertical
e^2	-	Ellipsoid eccentricity squared
a, b	-	Ellipsoid semi-major and semi-minor axes

The closed-form solution for the general inverse problem (which is the problem here) involves the solution of a quadratic equation and is not typically used in practice. Instead, an iterative solution is used for latitude and height for points that do not lie on the ellipsoid surface, i.e., for h ≠ 0.

To convert ECEF Cartesian coordinates to spherical coordinates:

Define:

$$\text{radius} = \sqrt{X^2 + Y^2 + Z^2}$$

$$\varphi' = \sin^{-1}\left(\frac{Z}{\text{radius}}\right)$$

$$\lambda = \tan^{-1}\left(\frac{Y}{X}\right)$$

Initialize:

$$\theta = \varphi'$$

$$h_0 = 0$$

Iterate until abs(h_i-h_{i+1}) < TOL

$$\text{re} = \frac{a * \sqrt{1-e}}{\sqrt{1-e * \cos^2(\theta)}}$$

$$\varphi = \tan^{-1}\left(\frac{\tan(\theta)}{1-e}\right)$$

$$\Delta\varphi = \varphi - \theta$$

$$\text{rs} = \text{radius}^2 - \text{re}^2 * \sin^2(\Delta\varphi)$$

$$\text{h}_{i+1} = \sqrt{\text{rs}} - \text{re} * \cos(\Delta\varphi)$$

$$\theta = \varphi' - \sin^{-1}\left(\frac{\text{h}_{i+1}}{\text{radius} * \sin(\Delta\varphi)}\right)$$

Projection Transformation

Convert coordinates from one map projection to another. The transformation from geodetic coordinates to the output map projection depends on the type of projection selected. The mathematics for the forward and inverse transformations for the Universal Transverse Mercator (UTM), Lambert Conformal Conic, Transverse Mercator, Oblique Mercator, Polyconic, and the Space Oblique Mercator (SOM) map projections are handled by U.S Geological Survey's (USGS) General Cartographic Transformation Package (GCTP), which may be obtained at http://edcftp.cr.usgs.gov/pub/software/gctpc/.

Grid Structure Summary

Tables 1 and 2 below show the detailed contents of the geometric grid structure.

Geometric Grid Structure Contents
Satellite Number (8)
WRS Path
WRS Row (may be fractional)
Acquisition Type (Earth, Lunar, Stellar)
Scene Framing Information:
Frame Type: PROJBOX, MAXBOX, PATH_MAXBOX, LUNAR, or STELLAR
Projection Units (text): METERS, RADIANS, ARCSECONDS
Projection Code: GCTP integer code for UTM, SOM, etc...
Datum: WGS84
Spheroid: GCTP integer code = 12 (WGS84/GRS80)
UTM Zone: UTM zone number (or 0 if not UTM)
Map Projection Parameters: 15-element double array containing parameters
Corners: 4 by 2 array of projection coordinates for UL, LL, UR, and LR corners
Path Oriented Framing Information:
Center Point: latitude and longitude of WRS scene center
Projection Center: Map x/y of WRS scene center
Rotation Angle: Rotation (from true north) of the path frame (degrees)
Orientation Angle: Rotation (from grid north) of the path frame (degrees)
Active Image Areas: latitude and longitude (in degrees) of the four corners of the active image area (excluding leading and trailing SCA imagery) for each band
Grid Structure Information:
Number of SCAs
Number of Bands
Band List: array of band IDs included in grid
Array of band grid structures, one for each SCA in each band (see Table 2)

Table 1: Geometric Grid Structure Contents

Grid Structure Contents for Each SCA in Each Band
Band number
Grid cell size: number of image lines and samples in each grid cell
Grid cell scale: 1/lines per cell and 1/samples per cell
Pixel size: in projection units (usually meters)
Number of lines in output image
Number of samples in output image
Number of lines in grid (NL)
Number of samples in grid (NS)
Number of z-planes (NZ)
Index of zero-elevation z-plane

Z-plane spacing: elevation increment between z-planes
1D array of input line numbers corresponding to each grid row
1D array of input sample numbers corresponding to each grid column
3D array of output lines for each grid point (row-major order) (NS*NL*NZ)
3D array of output samples for each grid point (row-major order) (NS*NL*NZ)
Array of line c_0, c_1 even/odd offset coefficients (row-major order) (2*NS*NL)
Array of sample d_0, d_1 even/odd offset coefficients (row-major order) (2* NS*NL)
3D array of forward mapping (ils2ols) coefficient sets (NS*NL*NZ)
3D array of inverse mapping (ols2ils) coefficient sets (NS*NL*NZ)
3D array of line jitter sensitivity coefficient vectors (note 2) (3*NS*NL*NZ)
3D array of sample jitter sensitivity coefficient vectors (note 2) (3*NS*NL*NZ)
Degree of rough polynomial
Array of rough line polynomial coefficients ((degree+1)^2 * NZ values)
Array of rough sample polynomial coefficients ((degree+1)^2 * NZ values)

Table 2: Per Band Geometric Grid Structure Contents

Geometric Grid Size

To fully capture the potential variability of the 50 Hz attitude data that will be available within the LDCM ancillary data stream would require a grid spacing of 5 lines. This may be impractical. Fortunately, the OLI error budgets assumed that attitude variations at frequencies up to only 10 Hz would be corrected in the LOS model. Such variations can be captured by sampling at 20 Hz or higher. This corresponds to a grid spacing of 11-12 lines. The grid has been successfully tested down to a line sampling of 10 but this does make for a large grid structure. The inclusion of a high frequency jitter table in the OLI model and jitter sensitivity coefficients in the grid structure allow the grid to be less dense in the time (line) dimension. The baseline assumption is that attitude frequencies above 3 Hz will be relegated to the jitter table allowing the grid density to be reduced to 30 lines thus saving grid space even with the addition of the new jitter sensitivity fields.

7.2.2.8 Notes

Some additional background assumptions and notes include:

1. The NOVAS planetary ephemeris file provides the lunar ephemerides used to define the reference output space for lunar image processing. This file is in the original JPL format and is provided to the NOVAS routines as an input.
2. The TIRS implementation of the LOS projection grid will include another new feature that has also been applied to OLI as of version 3.4 of this algorithm – a set of sensitivity coefficients that map roll-pitch-yaw deviations to the corresponding line and sample differences, for each grid cell. These sensitivity coefficients will be used by the resampler to convert high frequency (per image line) attitude variations to line and sample adjustments. Modeling the high frequency deviations separately and correcting them in the resampler allows for a sparser and more manageable sized grid.

7.2.3 OLI Line-of-Sight Model Correction Algorithm

7.2.3.1 Background/Introduction

The line-of-sight (LOS) model correction algorithm uses the results of ground control point (GCP) measurements in an image that was systematically and terrain corrected using the original LOS model, to derive estimates for corrections to the model. Corrections to the spacecraft attitude and ephemeris are computed in a least squares procedure which minimizes the differences between the measured locations of the control points and their true ground locations. This procedure includes the detection and removal of outlier GCPs using a modified t-distribution test. These corrections are added to the LOS model structure to create a “precision” model for subsequent use by other geometric algorithms. Creating the precision model involves repeating some of the processing originally performed by the LOS model creation algorithm to incorporate the model corrections. Once the precision model corrections are computed the algorithm performs simple threshold tests (e.g., on the pre-fit and post-fit RMS GCP residuals and the percentage of GCPs declared outliers) to determine if the solution was successful. If the solution is not successful, the LOS model is not updated with the corrections.

The OLI LOS model correction algorithm is derived from the ALI precision correction algorithm used in ALIAS. Its implementation should be very similar to the aliprecision application. As with the LOS model creation algorithm, model correction makes extensive use of the ALIAS geometric sensor (axx) and spacecraft (exx) model libraries.

7.2.3.2 Dependencies

The LOS Model Correction algorithm assumes that ground control points exist for the ground site and that the Model Creation, LOS Projection and Gridding, and Image Resampling algorithms have been executed to create a systematic terrain corrected image for GCP mensuration. Note that the band selection and resolution of this mensuration image will depend upon the flow being executed/control source being used. For standard L1T product generation the GLS control (SWIR1 band, 30m resolution) will be used whereas for characterization and calibration flows the DOQ control (panchromatic 15m) will be used. It further assumes that the GCP Correlation algorithm/utility has been executed to measure the GCP locations in the mensuration image. The mensuration image may be either SCA-separated or SCA-combined though SCA-combined images will be the preferred mode of operation.

7.2.3.3 Inputs

The LOS Model Correction algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data; including data set IDs to provide unique identifiers for data trending).

Algorithm Inputs
ODL File (implementation)
Measured GCP file name
OLI LOS model file name
OLI grid file name
DEM file name
CPF file name
L1G image file name
Precision solution parameters:
Apriori weights for attitude correction parameters (in microradians and microradians/second)

Apriori weights for ephemeris correction parameters (in meters and meters/second)
Correction model parameterization options (att_orb, eph_yaw, both, weight - default is "both")
Bias correction or rate of change correction option (time flag)
Apriori weights for GCP measurements (in at-sensor microradians)
Iteration limit
Outlier threshold
Processing Options (implementation):
Residual Trending On/Off Switch (new)
Solution/Alignment Trending On/Off Switch (new)
L0Rp ID (for trending)
Work Order ID (for trending)
Measured GCP File Contents (see GCP Correlation ADD for additional details)
GCP image positions
GCP ground coordinates
OLI Grid File Contents (see LOS Projection ADD for additional details)
Arrays of Input/Output Mappings
Output Image Frame (e.g., corners, map projection)
OLI LOS Model Contents (see LOS Model Creation ADD for additional details)
WRS Path/Row
Number of input image (L1R) lines
Smoothed image time codes
Integration Time (pan and multispectral bands) (new)
Smoothed ephemeris at 1 second intervals
Earth orientation parameters (UT1UTC, pole wander)
OLI to ACS reference alignment matrix/quaternion
Spacecraft CM to OLI offset in ACS reference frame (new)
Focal plane model parameters (number of SCAs, number detectors/band, Legendre coefficients)
Detector offset table (including detector deselect offsets) (new)
CPF File Contents
Pre-fit RMS threshold
Post-fit RMS threshold
Percent outlier threshold
Minimum number of valid GCPs threshold
L1G Image File Contents
L1G Metadata
DEM File Contents (see Maturity items 5 and 6)
DEM Metadata (new)
Elevation Data (new)

7.2.3.4 Outputs

Precision LOS Model (only items that are updated from the input LOS model are listed below)
Updated corrected ephemeris at 1 second intervals
Updated corrected attitude data sequence
Precision correction reference date/time
Precision attitude and ephemeris corrections
LOS Model Correction Solution File (see Table 1 below for additional details)
LOS model correction reference date/time
Final iteration precision correction values
Final iteration precision correction covariance
LOS Model Correction Residuals File (see Table 2 below for additional details)

GCP residuals for each point for each iteration
Correction Solution and Alignment Trending Data (new) (see Table 3 below for additional details)
Precision correction reference date/time
Precision attitude/ephemeris correction values (see note 1)
Reduced precision correction covariance (see note 2)
Solution quality metrics (see note 4)
Control type used (GLS or DOQ)
Off-nadir angle (in degrees)
L0Rp ID
Work Order ID
WRS Path/Row
Correction Residuals Trending Data (see note 3) (new) (see Table 4 below for additional details)
WRS Path/Row
GCP ID
GCP Type (GLS or DOQ)
Date/Time of imaging
Spacecraft position/velocity at image time
GCP ground coordinates (lat,lon,height)
Apparent GCP position (lat, lon, height) in mensuration image
LOS Model Correction Success/Failure Status Return (new)

7.2.3.5 Options

Solution/Alignment Trending On/Off Switch
Residual Trending On/Off Switch

7.2.3.6 Procedure

The LOS correction procedure uses the ground control point (GCP) measurements collected by the GCP Correlation algorithm to estimate updates to the spacecraft attitude and ephemeris data which minimize the discrepancies between the actual (known) GCP locations and the apparent locations measured in the terrain corrected L1G image. The solution method adopted for OLI is essentially the same as that used for Landsat 7 and for the ALI wherein "truth" and "observed" line of sight (LOS) vectors are constructed in the orbital coordinate system and a weighted least squares solution is used to minimize the misalignments between the truth and observed vectors. The solution supports the estimation of offset and rate corrections for all three ephemeris position axes and for all three attitude angles (roll-pitch-yaw) though options are provided to reduce the number of parameters (e.g., solve for offsets only) to accommodate situations where the ground control points are few in number, poorly distributed, or inaccurate.

There are several differences in the implementation of the OLI LOS correction model as compared to the previous missions. The first is a change in the coordinate system in which the corrections are applied. For Landsat 7 and ALI data, the precision corrections were applied in the orbital coordinate system. For OLI, they are applied in the spacecraft body/attitude control system coordinate system (this was also noted in the Ancillary Data Preprocessing ADD). For nadir-viewing scenes there is little difference but the case of off-nadir viewing leads to a few adjustments to the heritage algorithm in what follows.

The second significant difference is in the way that the corrections are reflected in the precision LOS model created as an output by this procedure. In the heritage implementation, the ephemeris corrections were used to update the model ephemeris data sequence but the attitude corrections

were stored as a separate correction model that was applied explicitly in the forward model. For OLI, corrected versions of both the ephemeris and attitude data sequences are computed using the LOS correction solution results. These corrected data are stored in the LOS model along with the original ephemeris and attitude values. The parameters of the correction model are also included in the model though they are there primarily for documentation purposes and are no longer used in the forward model computation.

The third difference is the inclusion of a portion of the sensor alignment calibration logic into the LOS correction algorithm. This logic uses the OLI to ACS alignment matrix stored in the OLI LOS model to convert the computed attitude offset corrections to OLI alignment angles. This yields updated estimates of the OLI to ACS alignment angles that are output to the characterization database for subsequent trending in the sensor alignment calibration procedure.

A fourth difference is the use of L1G terrain but not precision corrected images to measure the control points (see the GCP Correlation ADD). This gives the apparent (measured) GCP location a non-zero height coordinate. The true GCP elevation (from the known GCP ground location) could be used but it is more correct to interpolate the apparent point height from the DEM used to create the terrain corrected L1G mensuration image. The use of terrain corrected mensuration images also allows these images to be SCA-combined since the SCA overlap areas will be geometrically consistent.

The mathematical underpinnings of the LOS correction algorithm are presented first, followed by an overview of the procedure for implementing the algorithm.

Mathematical Development
The mathematical background of the LOS correction algorithm is presented in the following sub-sections. In what follows the equations presented are numbered so that they can be more easily referenced in the subsequent mathematical formulation and in the algorithm procedure sections.

1. Formulating the Observations
The geometric measurement in the OLI sensor system can be regarded as the look vector, $\boldsymbol{l}_{sc}$, in the spacecraft body-fixed system. This vector is transformed into the Orbit Reference Frame (OB) system (see Figure 1) as described in the Ancillary Data Preprocessing ADD, through the spacecraft attitude parameters:

$$\boldsymbol{l}_{ob} = \mathbf{T}^{T}(\theta_r, \theta_p, \theta_y)\, \boldsymbol{l}_{sc}. \qquad (1.1)$$

where θ_r, θ_p, and θ_y are roll, pitch, and yaw angles, **T** is the transformation matrix, and can be expressed as

$$\mathbf{T}(\theta_r, \theta_p, \theta_y) = \mathbf{R}_3(\theta_y)\mathbf{R}_2(\theta_p)\mathbf{R}_1(\theta_r)$$

$$= \begin{bmatrix} \cos\theta_p\cos\theta_y & \cos\theta_r\sin\theta_y + \sin\theta_r\sin\theta_p\cos\theta_y & \sin\theta_r\sin\theta_y - \cos\theta_r\sin\theta_p\cos\theta_y \\ -\cos\theta_p\sin\theta_y & \cos\theta_r\cos\theta_y - \sin\theta_r\sin\theta_p\sin\theta_y & \sin\theta_r\cos\theta_y + \cos\theta_r\sin\theta_p\sin\theta_y \\ \sin\theta_p & -\sin\theta_r\cos\theta_p & \cos\theta_r\cos\theta_p \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \theta_y & -\theta_p \\ -\theta_y & 1 & \theta_r \\ \theta_p & -\theta_r & 1 \end{bmatrix} \qquad (1.2)$$

where $\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{R}_3$ are the coordinate system transformation matrix for rotation around x, y and z-axis respectively.

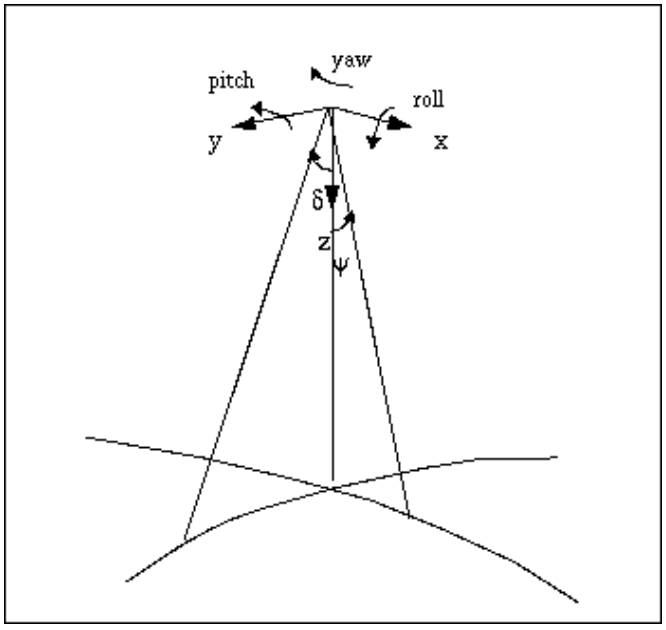


Figure 1: Definition of Orbit Reference System

The vector $\boldsymbol{l}_{ob}$ is further transformed into the ECF system

$$\boldsymbol{l}_{ef} = \mathbf{T}_f(\mathbf{r}_{ef}, \mathbf{v}_{ef})\, \boldsymbol{l}_{ob} \qquad (1.3)$$

where $\mathbf{T}_f$ is the forward transformation for vectors from the OB system to the ECF system, as a function of the satellite position $\mathbf{r}_{ef}$ and velocity $\mathbf{v}_{ef}$ vectors in the ECF system. Note that $\mathbf{v}_{ef}$ should be the "inertial" velocity expressed in the ECF system as described in the Ancillary Data Preprocessing ADD. Vector $\boldsymbol{l}_{ef}$, together with the satellite position vector, $\mathbf{r}_{ef}$, is then used to intersect the ellipsoid Earth surface to pin down a point position, $\mathbf{R}_{ef}$, as the target point on the Earth. This is the common forward image pixel geolocation calculation (forward model). Note that when using a terrain corrected L1G mensuration image, the $\mathbf{R}_{ef}$ point represents the intersection of the LOS with the DEM used to create the L1G image rather than the Earth ellipsoid surface. The target point will thus have a non-zero height coordinate.

Mathematically, $\mathbf{R}_{ef}$ is a function of $\boldsymbol{l}_{sc}$, θ_r, θ_p, θ_y, $\mathbf{r}_{ef}$, and $\mathbf{v}_{ef}$.

$$\mathbf{R}_{ef} = F(\boldsymbol{l}_{sc}, \theta_r, \theta_p, \theta_y, \mathbf{r}_{ef}, \mathbf{v}_{ef}) \qquad (1.4)$$

Because of errors in the satellite orbit ephemeris and attitude data, this calculated $\mathbf{R}_{ef}$ is different than the true location of the image pixel. If we know the true location of a landmark pixel ($\mathbf{R}_{cp}$) from other

sources (*i.e.*, base map, survey etc.), this point can be taken as a GCP to check the accuracy of the computed image pixel location. The precision correction process uses the GCP coordinates to estimate the correction to the satellite ephemeris and attitude data, so that with the corrected parameters in equation (1.4) the calculated image pixel location, $\mathbf{R}_{ef}$, will be close to its true location, $\mathbf{R}_{cp}$ (depending on the GCP positional accuracy).

To calculate the precision correction, the difference between $\mathbf{R}_{ef}$ and $\mathbf{R}_{cp}$ is taken as the observable, and the observation equation becomes:

$$d\mathbf{R} = \mathbf{R}_{cp} - F(\boldsymbol{l}_{sc}, \theta_r, \theta_p, \theta_y, \mathbf{r}_{ef}, \mathbf{v}_{ef}) \qquad (1.5)$$

according to equation (1.4). However, the actual calculation of $\mathbf{R}_{ef}$ is usually not an explicit function of the orbit and attitude parameters, especially for the intersecting procedure. Therefore, it is inconvenient to linearize equation (1.5) with standard estimation techniques. Instead, the calculation of look vector $\boldsymbol{l}_{cp}$ corresponding to $\mathbf{R}_{cp}$, in the OB system, is much more explicit:

$$\mathbf{l}_{cp} = \mathbf{T}_i(\mathbf{r}_{ef}, \mathbf{v}_{ef}) \frac{(\mathbf{R}_{cp} - \mathbf{r}_{ef})}{|\mathbf{R}_{cp} - \mathbf{r}_{ef}|} \qquad (1.6)$$

where $(\mathbf{R}_{cp} - \mathbf{r}_{ef})$ is the LOS vector in the ECF system corresponding to $\mathbf{R}_{cp}$, and $\mathbf{T}_i(\mathbf{r}_{ef}, \mathbf{v}_{ef})$ is the inverse transformation for the look vector from the ECF system to the OB system. If all the satellite attitude and ephemeris parameters are accurate, the $\boldsymbol{l}_{cp}$ from equation (1.6) and $\boldsymbol{l}_{ob}$ from equation (1.1) should be equal. Since the measurement $\boldsymbol{l}_{sc}$ is accurate compared to the attitude and ephemeris information, any systematic difference between $\boldsymbol{l}_{cp}$ and $\boldsymbol{l}_{ob}$ can be attributed to the attitude and orbit errors. Thus we can use the difference between $\boldsymbol{l}_{cp}$ and $\boldsymbol{l}_{ob}$ as the observable.

$$d\mathbf{l} = \boldsymbol{l}_{cp} - \boldsymbol{l}_{ob} = \mathbf{T}_i(\mathbf{r}_{ef}, \mathbf{v}_{ef}) \frac{(\mathbf{R}_{cp} - \mathbf{r}_{ef})}{|\mathbf{R}_{cp} - \mathbf{r}_{ef}|} - \mathbf{T}(\theta_r, \theta_p, \theta_y)\, \boldsymbol{l}_{sc} \qquad (1.7)$$

The task of precision modeling is then to calculate the correction to those satellite ephemeris and attitude parameters (*i.e.*, $\mathbf{r}_{ef}$, $\mathbf{v}_{ef}$ and θ's) so that the residuals of $d\mathbf{l}$ after correction are minimized for all selected GCPs. The orbit correction is modeled as a linear function of time for each component in the OB system. Referred to as the short arc method, this purely geometric method shifts and rotates the short arc of orbit defined by the original ephemeris points to fit the GCP measurements.

2. Linearizing the Observations

These observation equations can be linearized with the following steps. In equation (1.7), the calculation of $\boldsymbol{l}_{ob}$ can also be carried out through

$$\boldsymbol{l}_{ob} = T_i(\mathbf{r}_{ef}, \mathbf{v}_{ef})(\mathbf{R}_{ef} - \mathbf{r}_{ef}) / |\mathbf{R}_{ef} - \mathbf{r}_{ef}| \qquad (2.1)$$

if $\mathbf{R}_{ef}$ is more conveniently accessible. Since equation (2.1) is simply the inverse of equation (1.4) and equation (1.3), the $\boldsymbol{l}_{ob}$ calculated from equation (2.1) is the same as the one in equation (1.1) except for the possible inclusion of numerical errors. However, it should be mentioned that the true relationship between $\boldsymbol{l}_{ob}$ and the parameters is always equation (1.1). Equation (2.1) should not be confused with this because $\mathbf{R}_{ef}$ in equation (2.1) is not an independent variable but a function of

equation (1.4). So, in observation (1.7) information about the attitude parameters is contained in $\boldsymbol{l}_{ob}$ and the information about orbit parameters comes from $\boldsymbol{l}_{cp}$.

Since the measurement of $\boldsymbol{l}_{sc}$ is 2-dimensional in nature, only 2-dimensional information is contained in equation (1.7) though there are 3 components involved. If a look vector (either $\boldsymbol{l}_{cp}$ or $\boldsymbol{l}_{ob}$) has the three components in the OB system.

$$\mathbf{l} = \{xl, yl, zl\} \qquad (2.2)$$

The real information in these three components can be summarized in two variables like the original look angle measurements. We chose the following two variables:

$$\delta = \text{atan}\,(yl / zl) \qquad (2.3)$$

$$\psi = \text{atan}\,(xl / zl) \qquad (2.4)$$

So that the three components of equation (1.7) can be reduced to the two equations:

$$\alpha = \delta_{cp} - \delta_{ob} \qquad (2.5)$$

$$\beta = \psi_{cp} - \psi_{ob} \qquad (2.6)$$

Note that in equation (2.3) and (2.4) the components of *xl, yl,* and *zl* can be that of LOS vector instead of unit look vector, so that δ_{cp} and ψ_{cp} are explicit functions of orbit position. In that case *zl* is approximately the height of the satellite.

If we define,

true value = approximate value + correction

and differentiate equations (2.3) and (2.4) with respect to the orbit position (for δ_{cp} and ψ_{cp}), differentiate equation (1.1) with respect to the satellite attitude (for δ_{ob} and ψ_{ob}) at their corresponding approximate values, then equations (2.5) and (2.6) can be linearized as the function of correction parameters.

$$\alpha \cong (\cos^2\delta_{cp} / h)\, dy - (\cos\delta_{cp} \sin\delta_{cp} / h)\, dz + d\theta_r \qquad (2.7)$$

$$\beta \cong (1.0 / h)\, dx - d\theta_p + \tan\delta_{cp}\, d\theta_y \qquad (2.8)$$

where dx, dy, and dz are the correction to satellite position vector $\mathbf{r}_{ob}$ in the OB system, and $d\theta$'s are the corrections to the satellite attitude angle θ's. Other quantities are functions evaluated at the approximate values of $\mathbf{r}_{ef}$, $\mathbf{v}_{ef}$, and θ's.

The linearization above is done by directly differentiating equation (2.3) and (2.4), with transformation $\mathbf{T}_i$ regarded unaffected by the error in $\mathbf{r}_{ef}$ and $\mathbf{v}_{ef}$. This, however, ignores the curvature of the satellite orbit and the Earth, resulting in about 10% of error in the coefficients of dx, dy, and dz. A more accurate way to evaluate these coefficients is to examine the sensitivity terms $d\psi_{cp}/dx$, $d\delta_{cp}/dy$, and $d\delta_{cp}/dz$ through the geometry of the look vector (see Figure 2).

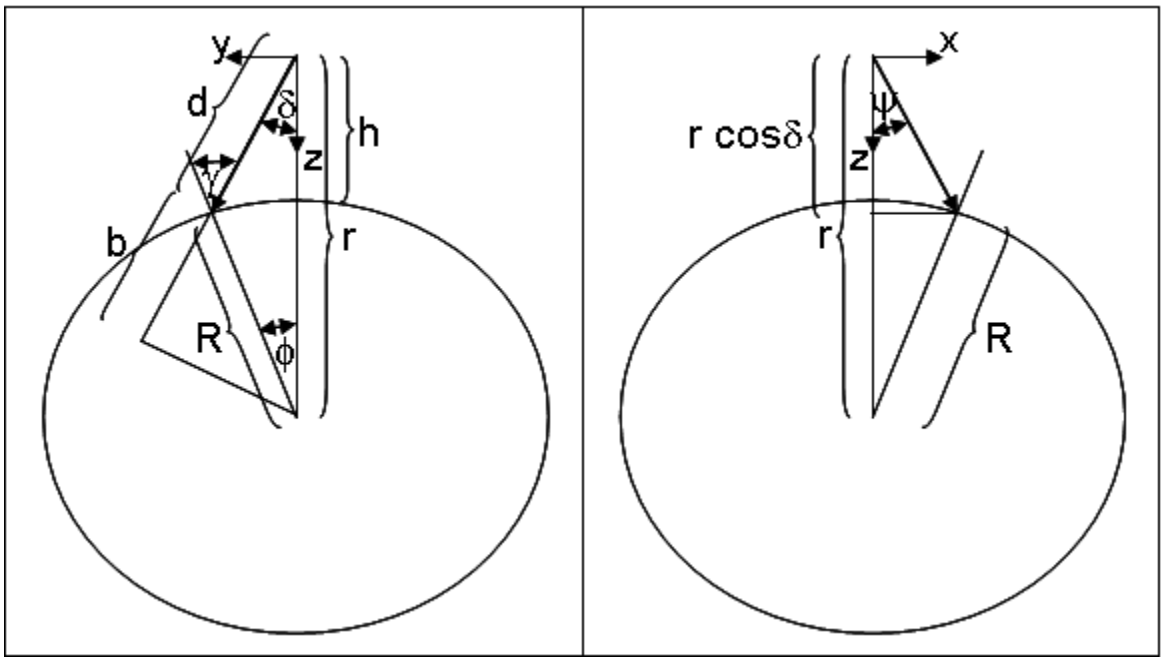


Figure 2: Look Vector Geometry

R – the radius of the Earth
r – the radius of the satellite position
h – the altitude of the satellite
d – the magnitude of the look vector (from satellite to target)
δ – the across-track angle of the look vector
ϕ – the Earth centered angle between the satellite and the target
γ – the zenith angle of the look vector at the target
x,y,z – the coordinates of the satellite position in the OB system

We have

$$R \sin(\delta + \phi) = r \sin \delta \qquad (2.9)$$

Differentiating the equation (holding R and r constant) yield

$$R \cos(\delta + \phi)(d\delta + d\phi) = r \cos\delta \, d\delta \qquad (2.10)$$

Note that when $\delta + \phi = \gamma$, and $d\phi = -dy / r$, we have

$$\alpha = d\delta = (-b / (r\, d))\, dy \qquad (2.11)$$

Similarly for the along-track direction, we have

$$\beta = d\psi = (-(r - d \cos\delta) / (r\, d \cos\delta))\, dx \qquad (2.12)$$

For the effect of altitude error, differentiate equation (2.9) with respect to δ and *r* (holding ϕ constant) and noting dr = -dz, we have

$$\alpha = d\delta = (\sin\delta / d)\, dz \qquad (2.13)$$

Note that the dx, dy, and dz in equations (2.11) through (2.13) are error terms, which are opposite in sign to the correction terms. With this in mind, we can replace the correction terms in equation (2.7) and (2.8) and rewrite the linearized observation equation as:

$$\alpha = (b / (r\, d))\, dy - (\sin\delta / d)\, dz + d\theta_r \qquad (2.14)$$

$$\beta = ((r - d\cos\delta) / (rd\cos\delta))\, dx - d\theta_p + \tan\delta\, d\theta_y \qquad (2.15)$$

where:

$$b = R\cos\gamma = \mathrm{sqrt}(R^2 - (r^2\sin^2\delta)) \qquad (2.16)$$

$$d = r\cos\delta - b \qquad (2.17)$$

This formulation does not account for the effects of applying the attitude correction in the ACS/body frame rather than the orbital frame. This is particularly significant in the case of off-nadir pointing. In the general case, applying the attitude correction in the ACS coordinate system leads to the following linearized observation equations:

$$\alpha = (b / (r\, d))\, dy - (\sin\delta / d)\, dz + M_{11}\, d\theta_r + M_{12}\, d\theta_p + M_{13}\, d\theta_y \qquad (2.18)$$

$$\beta = ((r - d\cos\delta) / (rd\cos\delta))\, dx + (M_{31}\tan\delta - M_{21})\, d\theta_r + (M_{32}\tan\delta - M_{22})\, d\theta_p + (M_{33}\tan\delta - M_{23})\, d\theta_y \qquad (2.19)$$

where:

M_{11}, M_{12}, M_{13}, M_{21}, M_{22}, M_{23}, M_{31}, M_{32}, M_{33} are the elements of the ACS to Orbital rotation matrix $\mathbf{M}_{ACS2ORB}$ at the time of the GCP observation. Thus, it is necessary to know the spacecraft roll-pitch-yaw corresponding to the GCP.

$\mathbf{M}_{ACS2ORB}$ =

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y) + \cos(r)\sin(y) & \sin(r)\sin(y) - \cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y) - \sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y) + \sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

Note that for nominal nadir viewing $M_{11} = M_{22} = M_{33} = 1$ and $M_{12} = M_{13} = M_{21} = M_{23} = M_{31} = M_{32} = 0$ and equations (2.18) and (2.19) reduce to equations (2.14) and (2.15).

Both linearized observation equations (2.18) and (2.19) include all three attitude correction terms. This has the effect of linking the along- and across-track observations in the new OLI formulation, unlike the heritage implementation which used separate along- and across-track solutions.

3. Weighted Least Squares Solution

A weighted least squares solution to the parameters is found using the following steps. The correction parameters in equations (2.18) and (2.19) can be expanded to include the correction to the change rates of the satellite attitude and position by defining

$$dx = dx_0 + dx_{dot}dt \quad \text{and} \quad d\theta_r = d\theta_{r0} + d\theta_{rdot}dt \tag{3.1}$$

Since both the coordinates of the GCP and the measurement of the apparent GCP location in the image contribute random errors in computing α and β, the covariance matrix for the observation equations (2.18) and (2.19) should be the sum of the covariance matrix of $\mathbf{R}_{cp}$ in equation (1.6) and the covariance matrix of $\mathbf{R}_{ef}$ in equation (2.1), mapped through equations (1.7), (2.3), and (2.4).

Note that in the observation equations (2.14) / (2.15) and (2.18a) / (2.19a), α is only related to parameters dy, dz, and $d\theta_r$, and β is only related to dx, $d\theta_p$, and $d\theta_y$. The parameters are uncoupled in the two observations. In the simplified case where observational error of α and β are uncorrelated, the observation equations can be separated into two independent equations and solved individually. In the more general case of equations (2.18) and (2.19) the equations are coupled and must be solved together. This coupling is, in fact, present due to the yaw offsets introduced by yaw steering.

While it might be tempting to try to circumvent this complication by redefining the orbital coordinate system to be based on the Earth-rotation corrected ECEF velocity vector (thereby "yaw-steering" the orbital coordinate system) this would lead to a different set of complications in the application of the ephemeris corrections. In the baseline algorithm we will adopt the general formulation of equations (2.18) and (2.19) and have adjusted the heritage separable least squares solution formulation accordingly.

Proceeding with the integrated formulation, we define the parameter vector as:

$$\mathbf{X}' = \{d\theta_{r0}, d\theta_{p0}, d\theta_{y0}, dx_0, dy_0, dz_0, d\theta_{rdot}, d\theta_{pdot}, d\theta_{ydot}, dx_{dot}, dy_{dot}, dz_{dot}\} \tag{3.2}$$

(deleted) (3.3)

where ' means transpose of a vector or matrix. Then, the two observation equations can be written as:

$$\alpha = h_1 \mathbf{X} + \varepsilon_a \tag{3.4}$$

$$\beta = h_2 \mathbf{X} + \varepsilon_b \tag{3.5}$$

where:

$$h_1 = \{ M_{11}, \; M_{12}, \; M_{13}, \; 0.0, \; b/(d\,r), \; -\sin\delta/d, \; M_{11}\,dt, \; M_{12}\,dt, \; M_{13}\,dt, \; 0.0, \; b\,dt/(d\,r), \; -\sin\delta\,dt/d\} \tag{3.6}$$

$$h_2 = \{(M_{31}\tan\delta - M_{21}), \; (M_{32}\tan\delta - M_{22}), \; (M_{33}\tan\delta - M_{23}), \; (r - d\cos\delta)/(r\,d\cos\delta), \; 0.0, \; 0.0, \; (M_{31}\tan\delta - M_{21})\,dt, \; (M_{32}\tan\delta - M_{22})\,dt, \; (M_{33}\tan\delta - M_{23})\,dt, \; (r - d\cos\delta)\,dt/(r\,d\cos\delta), \; 0.0, \; 0.0\} \tag{3.7}$$

with M_{11}, M_{12}, M_{13}, M_{21}, M_{22}, M_{23}, M_{31}, M_{32}, M_{33} the elements of the ACS to Orbital rotation matrix $\mathbf{M}_{ACS2ORB}$ at the time of the GCP observation.

ε_a and ε_b are the random error of α and β respectively. With all GCPs included, the along- and across-track observation equation can be written as:

$$\boldsymbol{A} = H_1\mathbf{X} + \varepsilon_A \qquad (3.8)$$

$$\boldsymbol{B} = H_2\mathbf{X} + \varepsilon_B \qquad (3.9)$$

and the integrated parameters can be solved by WLS estimation as:

$$\mathbf{X} = (H_1'WaH_1 + H_2'WbH_2)^{-1}(H_1'Wa\boldsymbol{A} + H_2'Wb\boldsymbol{B}) \qquad (3.10)$$

where ***A*** and ***B*** are the observation vectors, composed of α and β for all the GCPs, respectively. H_1 and H_2 are corresponding coefficient matrix with h_1 and h_2 as rows corresponding to each α and β, *Wa* and *Wb* are the diagonal weight matrix for ***A*** and ***B*** respectively, composed of inverse of the variance of each individual ε_a and ε_b.

4. Parameter Correlation and Covariance Estimation

One problem in this solution is the nearly linear correlation between parameter dx and $d\theta_p$ in the observation equation (3.7). The along-track orbit error and the pitch angle error have the very similar effect on β. The two parameters cannot be well separated in the solution without additional information – including both parameters in the observation equations results in a near-singular normal equation and therefore an unstable solution of the parameters. Similarly, high correlation exists between the cross-track position and the roll attitude errors in equation (3.6) and an ill-conditioned normal equation would result.

For the purpose of correcting the image, we do not have to distinguish between orbit position correction and attitude correction parameters. Letting either the orbit or attitude correction parameters absorb the existing errors will correct the image in a similar manner. Therefore, we can choose to estimate either dx and dy or $d\theta_p$ and $d\theta_r$. This can be done by setting those coefficients in h_1 and h_2, corresponding to the unwanted parameters to zero.

One of the challenging tasks is to distinguish satellite attitude error from the orbit positional error. The purpose of precision correction estimation is not only to correct the image but also to extract information about the sensor alignment, which is reflected in the attitude correction parameters. In order to separate the ephemeris error from the attitude error as much as possible, we should first use the most precise ephemeris data available and correct systematic errors with available models. Second we should use available a priori information in addition to the observation to cure the ill condition of the normal equation in statistical estimation.

Let the observation equation be:

$$\boldsymbol{Y} = H\boldsymbol{X} + \varepsilon;$$
$$E[\varepsilon] = 0, \quad Cov[\varepsilon] = s^2C \qquad (4.1)$$

where ***Y*** is the measurement vector, ***X*** the parameter vector, *H* the coefficient matrix and ε the residual error vector, and s^2 is a covariance scaling factor;

and the a priori information of the parameters be:

$$\boldsymbol{X}\_ = \boldsymbol{X} + \varepsilon_x;$$
$$E[\varepsilon_x] = 0, \ \ Cov[\varepsilon_x] \ = q^2 C_x \qquad (4.2)$$

where $\boldsymbol{X}\_$ is the apriori parameter vector, ε_x is the residual vector and q^2 is a covariance scaling factor;

then the normal equation for the Best Linear Unbiased Estimate (BLUE) $\boldsymbol{X}\hat{}$ of the unknown parameter vector ***X*** is:

$$((I/s^2)H'WH + (I/q^2)W_x)\boldsymbol{X}\hat{} = (I/s^2)H'W\boldsymbol{Y} + (I/q^2)W_x\boldsymbol{X}\_ \qquad (4.3)$$

where *W* and W_x are weight matrices.

$$W = C^{-1};$$

$$W_x = C_x^{-1} \qquad (4.4)$$

The covariance matrix of $\boldsymbol{X}\hat{}$ is:

$$Cov[\boldsymbol{X}\hat{}] = ((I/s^2)H'WH + (I/q^2)W_x)^{-1} \qquad (4.5)$$

Usually, the Cov[ε] and Cov[ε_x] can not be exactly known. In the case of GCP, for example, the position error involves many factors like base map error, and human marking error, etc... If there are unknown scale factors s^2 and q^2, we can still obtain the WLS estimate from the normal equation.

$$(H'WH + W_x)\boldsymbol{X}\hat{} = H'W\boldsymbol{Y} + W_x\boldsymbol{X}\_ \qquad (4.6)$$

In such case, the inverse of the normal matrix can not be taken directly as the Cov[$\boldsymbol{X}\hat{}$]. Factor s^2 and q^2 should be estimated with appropriate variance component estimation from the residual of the solution of equation (4.6). The weighted residual square summation can be calculated as:

$$V'W\boldsymbol{V} = \boldsymbol{Y}'W\boldsymbol{Y} - 2\,\boldsymbol{X}\hat{}'M + \boldsymbol{X}\hat{}'\,N\boldsymbol{X}\hat{} \qquad (4.7)$$

$$V_x'W_x\boldsymbol{V}_x = \boldsymbol{X}\_'W_x\boldsymbol{X}\_ - 2\,\boldsymbol{X}\hat{}'W_x\boldsymbol{X}\_ + \boldsymbol{X}\hat{}'W_x\boldsymbol{X}\hat{} \qquad (4.8)$$

where:

$V = \boldsymbol{Y} - H\boldsymbol{X}\hat{}$ the measurement residual vector (4.9)

$\boldsymbol{V}_x = \boldsymbol{X}\_ - \boldsymbol{X}\hat{}$ the apriori parameter residual vector (4.10)

$$N = H'WH \qquad (4.11)$$

$$M = H'W\boldsymbol{Y} \qquad (4.12)$$

When the factors s^2 and q^2 are appropriately estimated, the weight matrix $\boldsymbol{W}$ and $\boldsymbol{W_x}$ should be correspondingly corrected by factors $1/s^2$ and $1/q^2$, respectively. Equation (4.6) should be resolved with the new weight matrices. In the new solution, information from the observation and the a priori information are appropriately combined and the
$(H' WH + W_x)^{-1}$ is the Cov[$\boldsymbol{X}$^].

5. Weight Factor Estimation
One of the estimates of s^2 and q^2 is the Helmert type estimate. For the problem here, the equation for the estimate can be derived following Helmert's variance component analysis,

$$E\, s^2 + D\, q^2 = \boldsymbol{V'WV} \qquad (5.1)$$

$$D\, s^2 + G\, q^2 = \boldsymbol{V_x'}W_x\boldsymbol{V_x} \qquad (5.2)$$

where:

$$E = n - 2\, tr\{Q\, N\} + tr\{Q\, N\, Q\, N\} \qquad (5.3)$$

$$G = m - 2\, tr\{Q\, W_x\} + tr\{Q\, W_x\, Q\, W_x\} \qquad (5.4)$$

$$D = tr\{Q\, N\, Q\, Wx\} \qquad (5.5)$$

$$Q = (H'\, W\, H + W_x)^{-1} \qquad (5.6)$$

n = number of observations

m = number of parameters

tr{A} indicates the trace of matrix *A*

Equation (5.1) and (5.2) do not guarantee positive solution of s^2 and q^2. In some cases, especially for small s^2 and q^2, noise can drive the solution negative. Another type of estimate, the iterative Maximum Likelihood Estimate (MLH), guarantees positive solution, though the estimate s^2 and q^2 may not be statistically unbiased. The MLH solution is obtained by iteratively solving equation (4.6) and

$$s^2 = \boldsymbol{V'WV} / n \qquad (5.7)$$

$$q^2 = \boldsymbol{V_x'}W_x\boldsymbol{V_x} / m \qquad (5.8)$$

$$W = W / s^2 \qquad (5.9)$$

$$W_x = W_x / q^2 \qquad (5.10)$$

until s^2 and q^2 converge.

The solution above provides the estimate of the corrections to the ephemeris and attitude data as well as to their covariance matrix. The covariance information can be used as a measure of precision for assessing the alignment errors of the sensor system. It can also be propagated to any pixel in the scene to evaluate the pixel location error after the precision correction.

6. Covariance Propagation

Given the sample time and across-track look angle of a pixel, the coefficients h_1 and h_2 can be calculated for α and β according to equation (3.6) and (3.7). The variance of α and β are then calculated as:

$$\sigma_\alpha^2 = h_1 \, \mathrm{Cov}[\mathbf{X^\wedge}] h_1' \qquad (6.1)$$

$$\sigma_\beta^2 = h_2 \, \mathrm{Cov}[\mathbf{X^\wedge}] h_2' \qquad (6.2)$$

These are the variance of the pixel location in sample and line directions due to the uncertainty of the estimated precision correction parameters. They are in angles but can be easily converted into IFOV according to the sensor system specifications.

7. Outlier Detection

Outlier detection for the precision correction solutions seeks to identify GCPs that are likely to be in error due to miscorrelation. This is done by analyzing the GCP residuals, taking into account the relative importance of the GCP as reflected in the precision solution normal equation matrix.

Definitions:

- **A** = matrix of coefficients (partial derivatives) relating parameters to observations
- $\underline{\theta}$ = parameter vector
- $\underline{X}$ = observation vector
- $\underline{V}$ = residual vector
- **C** = observation covariance matrix
- n = the number of observations
- p = the number of parameters

A is n x p, $\underline{\theta}$ is p x 1, $\underline{X}$ and $\underline{V}$ are n x 1, and **C** is n x n

Observation Equation:

$$\mathbf{A}\underline{\theta} = \underline{X} - \underline{V} \qquad (7.1)$$

$$\underline{X} = \underline{X}_{true} + \underline{E} \quad \text{where E = error vector} \sim G(\underline{0}, \mathbf{C}) \qquad (7.2)$$

$$\mathbf{A}\underline{\theta}_{true} = \underline{X}_{true} \quad \text{where } \underline{\theta}_{true} \text{ is the “true” parameter vector} \qquad (7.3)$$

$$\mathbf{A}\underline{\theta} = \underline{X}_{true} + \underline{E} - \underline{V} \qquad (7.4)$$

so $\underline{V} = \underline{E}$ if $\underline{\theta} = \underline{\theta}_{true}$

Minimum Variance Parameter Estimate:

$$\underline{\theta}' = [\mathbf{A}^T\mathbf{C}^{-1}\mathbf{A}]^{-1}\mathbf{A}^T\mathbf{C}^{-1}\underline{X} \qquad (7.5)$$

Estimated Residual Error:

$$\underline{V}' = \underline{X} - \mathbf{A}[\mathbf{A}^T\mathbf{C}^{-1}\mathbf{A}]^{-1}\mathbf{A}^T\mathbf{C}^{-1}\underline{X} \quad (7.6)$$

Define Projection matrix **P**:

$$\mathbf{P} = \mathbf{A}[\mathbf{A}^T\mathbf{C}^{-1}\mathbf{A}]^{-1}\mathbf{A}^T\mathbf{C}^{-1} \quad (7.7)$$

This matrix projects the observation vector into the parameter subspace (the column space of **A**). This projection is only orthogonal if **C** has the special structure described below.

Substituting:

$$\underline{V}' = \underline{X} - \mathbf{P}\underline{X} = [\mathbf{I} - \mathbf{P}]\underline{X} \quad (7.8)$$

[**I** - **P**] projects $\underline{X}$ into the parameter null space.

Looking at the Error Estimate $\underline{V}'$:

$$\underline{V}' = [\mathbf{I} - \mathbf{P}]\underline{X} = [\mathbf{I} - \mathbf{P}][\underline{X}_{true} + \underline{E}] = [\mathbf{I} - \mathbf{P}]\underline{X}_{true} + [\mathbf{I} - \mathbf{P}]\underline{E} \quad (7.9)$$

but $[\mathbf{I} - \mathbf{P}]\underline{X}_{true} = \underline{0}$ since $\underline{X}_{true}$ lies entirely within the parameter subspace.

$$\text{so } \underline{V}' = [\mathbf{I} - \mathbf{P}]\underline{E} = \underline{E} - \mathbf{P}\underline{E} \quad (7.10)$$

Here are some comments about $\underline{V}'$ and $\underline{E}$:

- For a given precision solution the elements of $\underline{E}$ are not random variables, they are realizations of random variables.
- $\underline{V}'$ is an estimate of the actual (realized) error $\underline{E}$ which includes an estimation error equal to $\mathbf{P}\underline{E}$.
- We cannot exactly recover $\underline{E}$ from $[\mathbf{I} - \mathbf{P}]^{-1}\underline{V}'$ because [**I** - **P**] is singular (it is an n x n matrix of rank n-p).
- We can attempt to predict how accurate our estimate ($\underline{V}'$) of $\underline{E}$ is likely to be by looking at the estimation error $\underline{R} = \mathbf{P}\underline{E}$.
- Since we want the predicted accuracy to apply in general, we treat $\underline{R}$ as a random vector, which is a function of another random vector $\underline{E}$.

Expected Value: $E[\underline{R}] = E[\mathbf{P}\underline{E}] = \mathbf{P}\,E[\underline{E}] = \mathbf{P}\,\underline{0} = \underline{0}$ (7.11)

Variance: $E[\underline{R}\,\underline{R}^T] = E[\mathbf{P}\,\underline{E}\,\underline{E}^T\,\mathbf{P}^T] = \mathbf{P}\,E[\underline{E}\,\underline{E}^T]\,\mathbf{P}^T = \mathbf{P}\,\mathbf{C}\,\mathbf{P}^T$ (7.12)

Special Structure of Observation Covariance Matrix for Precision Correction:

$$\mathbf{C} = \sigma^2\mathbf{I} \quad (7.13)$$

since the observation errors are realizations of independent and identically distributed zero mean Gaussian random variables with variance σ^2.

Substituting (7.13) into equation (7.7) for **P** yields:

$$\mathbf{P} = \mathbf{A}[(1/\sigma^2)\mathbf{A}^T\mathbf{I}\mathbf{A}]^{-1}\mathbf{A}^T\mathbf{I}(1/\sigma^2) = \mathbf{A}\sigma^2[\mathbf{A}^T\mathbf{A}]^{-1}\mathbf{A}^T(1/\sigma^2) = \mathbf{A}[\mathbf{A}^T\mathbf{A}]^{-1}\mathbf{A}^T \quad (7.14)$$

And the equation for the variance of $\underline{R}$:

$$E[\underline{R}\,\underline{R}^T] = \sigma^2\,\mathbf{P}\,\mathbf{I}\,\mathbf{P}^T = \sigma^2\,\mathbf{P} \quad (7.15)$$

noting that $\mathbf{P}^T = \mathbf{P}$ and $\mathbf{P}\,\mathbf{P} = \mathbf{P}$

so $\underline{R} \sim G(\underline{0}, \sigma^2\,\mathbf{P})$

For a particular component of $\underline{R}$ r_i:

$$E[r_i] = 0 \quad (7.16)$$

$$E[r_i^2] = \sigma^2\,p_{ii} \quad (7.17)$$

Where p_{ii} is the i^{th} diagonal component of **P**

Looking at the equation for **P** we see that:

$p_{ii} = \underline{A}_i^T [\mathbf{A}^T\mathbf{A}]^{-1} \underline{A}_i$ (7.18)

Where $\underline{A}_i^T$ is the i^{th} row of **A**

Considering a particular component of the Residual Error Vector $\underline{V}$':

$v_i = e_i - r_i$ (7.19)

Where e_i is the corresponding component of the observation error vector so v_i is an unbiased estimate of e_i with variance $\sigma^2 p_{ii}$

If we knew what e_i was, we could test it against a probability threshold derived from its standard deviation, σ, to determine if it is likely to be an outlier. Instead of e_i we have v_i which includes the additional error term r_i. Including the additional estimation error in the threshold computation leads to:

$\sigma_v^2 = \sigma^2 + \sigma^2 p_{ii}$ (7.20)

Where σ^2 is the term due to the actual error variance and $\sigma^2 p_{ii}$ is the term due to the estimation error variance.

This may seem like cheating since e_i and r_i are not independent for a given realization.

$E[v_i^2] = E[(e_i - r_i)^2] = E[e_i^2 - 2e_ir_i + r_i^2]$ and $r_i = \Sigma_j p_{ij} e_j$

$E[v_i^2] = \sigma^2 (1 - p_{ii})$ (7.21)

It is tempting to use $v_i / (1 - p_{ii})^{1/2}$ for e_i in the outlier test (or, equivalently, to test v_i against a threshold based on $\sigma^2 (1 - p_{ii})$) but this becomes dangerous as p_{ii} approaches 1. The factor p_{ii} can be interpreted as a measure of the uniqueness of, or as the information content of, the i^{th} observation. As p_{ii} approaches 1, the i^{th} observation lies almost entirely within the parameter subspace, which implies that it is providing information to the solution that the other observations do not. Note that such "influential" observations can be identified from the structure of the coefficient matrix, **A**, without reference to the observation residuals. Attempting to use $1/(1 - p_{ii})^{1/2}$ to rescale the residual v_i to better approximate e_i will, in a sense, punish this observation for being important. Instead, we view p_{ii} as a measure of how poor an estimate of the actual error, e_i, the residual, v_i, is and ignore the fact that v_i will tend to be an underestimate of e_i. We therefore use σ_v^2 (= $\sigma^2 (1 + p_{ii})$ as shown above) to construct the outlier detection threshold.

One remaining problem is that we do not know exactly what σ^2 is and must estimate it from the observation residuals. This is done by scaling the a priori observation variance using the variance of unit weight that was computed in the precision solution. The fact that we are using an estimated variance to establish our outlier detection threshold modifies the algorithm in two ways: 1) we compensate for the fact that removing a point as an outlier will alter the computation of the variance of unit weight by removing one residual and reducing the number of degrees of freedom; and 2) we base the detection threshold computation on student's t-distribution rather than the Gaussian distribution.

The variance of unit weight is computed as:

$var_0 = \underline{V}^T\mathbf{C}^{-1}\underline{V}/(n - p) = \underline{V}^T\underline{V}/\sigma_0^2(n - p) = \Sigma v_j^2/\sigma_0^2(n - p)$ (7.22)

Where: n = number of observations,
p = number of parameters, and
σ_0^2 is the a priori variance.

The estimated variance is:

$$var = var_0\ \sigma_0^2 = \Sigma v_j^2/(n - p) \qquad (7.23)$$

Removing the k^{th} observation makes this:

$$var_k = (\Sigma v_j^2 - v_k^2)/(n - 1 - p) = (n - p)/(n - p - 1) * (\Sigma v_j^2 - v_k^2)/(n - p)$$

$$var_k = (n - p)/(n - p - 1) * var - v_k^2/(n - 1 - p) \qquad (7.24)$$

To normalize the k^{th} residual we divide it by the estimated standard deviation $\sigma' = (var)^{1/2}$:

$$w_k = v_k / \sigma' \qquad (7.25)$$

We can rescale this normalized residual to reflect the removal of this observation from the variance estimate without having to actually compute a new variance:

$$w_k' = v_k / \sigma_k' = w_k\ \sigma'/\sigma_k' = w_k\ (var/var_k)^{1/2}$$

$$var/var_k = 1 / [(n - p)/(n - p - 1) - v_k^2/var\ (n - p - 1)] = (n - p - 1)/(n - p - v_k^2/var)$$

$$var/var_k = (n - p - 1)/(n - p - w_k^2)$$

noting that $v_k^2/var = w_k^2$

$$w_k' = w_k\ [(n - p - 1)/(n - p - w_k^2)]^{1/2} \qquad (7.26)$$

Finally, we include the $(1 + p_{kk})$ factor discussed above and our normalized and reweighted residual becomes:

$$w_k' = w_k\ [(n - p - 1)/(1 + p_{kk})(n - p - w_k^2)]^{1/2} \qquad (7.27)$$

where: $w_k = v_k / \sigma'$

This normalized and reweighted residual is compared against a probability threshold computed using Student's t-distribution with (n - p) degrees of freedom.

LOS Correction Procedure Overview

The precision correction procedure developed mathematically above is implemented as an iterative solution to account for the non-linearity of the observation equations presented in (2.5) and (2.6) above. Each step in the iteration solves the linearized correction problem using equation (3.10) above, using the current correction estimates, to compute incremental corrections for the current iteration. These corrections are used to update the current estimates, and the iteration continues until the incremental corrections are smaller than some threshold (or the iteration limit is exceeded).

An additional layer of iteration is introduced by the need to perform outlier filtering on the input GCP data. The procedure thus includes two levels of iteration: 1) use the current active set of GCPs to perform the iterative weighted least squares solution (the linearization iteration); 2) filter the resulting GCP residuals for outliers, remove those exceeding the specified tolerance, and iterate the weighted least squares procedure with the new (reduced) active set until no new outliers are found.

The LOS correction procedure can be viewed as a five phase process in which the third and fourth phases are nested:

a) Phase 1 - Load the necessary data and initialize the solution procedure.
b) Phase 2 - Load and initialize the GCPs. For each GCP, use the geometric grid to compute the input space (L1R) location and time of observation. Interpolate the spacecraft position, velocity, and attitude at the time of observation.
c) Phase 3 - Use the current active set of GCPs to form and solve the linearized weighted least squares equation. Use the computed corrections to update the current estimates. Iterate the

linearized solution procedure until the incremental corrections are below the convergence threshold. Compute and write residuals for each iteration (the initial pre-correction and final iteration residuals are both used in geodetic accuracy assessment). Note that the residuals file is reinitialized at the beginning of each phase 4 loop so that the output residual file will reflect only the final pass through the outlier detection loop (phase 4).

d) Phase 4 - Run the outlier detection and removal iteration loop using the results of the iterative weighted least squares solution procedure (phase 3) by testing the resulting residuals for outliers. Remove any newly detected outliers from the active GCP list and recompute the phase 3 solution with the reduced GCP set. Continue to iterate until no new outliers are detected.
e) Phase 5 - Write the precision solution file to document the final result, update the LOS model, and, if requested, convert the attitude corrections to OLI alignment angles and write the resulting alignment calibration information to the characterization database.

Figure 3 shows a block diagram of the LOS correction procedure in which the individual process steps are identified by phase using the codes P1 through P5.

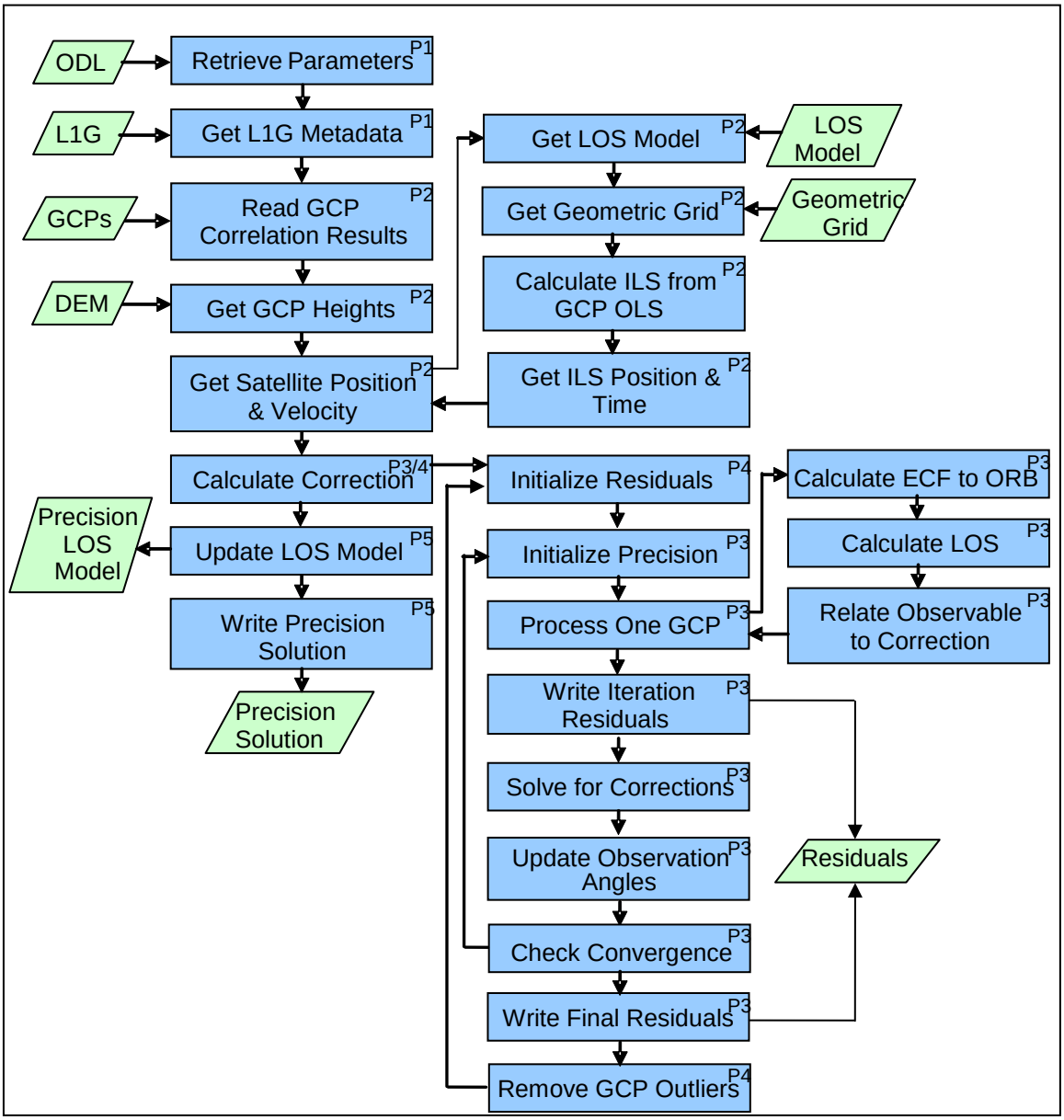


Figure 3: LOS Correction Algorithm Block Diagram

We next examine the individual process steps in the prototype LOS correction procedure.

7.2.3.7 Prototype Code

Inputs to the executable are an ODL parameter file, an ASCII GCP measurement file created by the GCP correlation algorithm, the L1G image used to measure the GCPs, the OLI LOS model used to create the L1G image, the LOS projection grid file used to create the L1G image, the calibration parameter file used to create the L1G image, and the DEM file (if any) used to terrain correct the L1G image. Note that only the L1G image metadata is used, not the imagery itself. The outputs are an updated (precision corrected) OLI LOS model file, an ASCII report file containing a standard header that identifies the data set analyzed, and the results of the precision correction solution, and an ASCII file containing the computed GCP residual errors at each iteration of the solution. The GCP residuals for the first and last iteration are subsequently used by the geodetic accuracy characterization

algorithm. The prototype implementation also includes an option to generate trending data which, rather than being stored in a database, is written to the standard output in a comma-delimited format.

The prototype code accesses two environment variables to populate fields used in the standard report header. These are IAS_REL which contains the IAS software version number, and IAS_SITE which contains a text string identifying the processing center.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall -march=nocona -m32

Get Precision Parameters (get_prec_par)

This function gets the precision correction algorithm parameters.

Read Precision Parameters (read_prec_parm)

This function reads all the parameters from ODL and CPF files that precision requires.

Get Position (get_position)

This function finds the satellite position, velocity, attitude and reference time for each GCP.

Add Position (add_position)

This function adds the position to the ground control point structure and assigns the reference time to the time structure using the following steps.

1. From the 1G line and sample, find the latitude and longitude, and use the DEM to find the height at that line/sample (new for LDCM); then transform to earth fixed.
2. From the L1G line and sample use the geometric grid and the ols2ils routine (reference the LOS Projection ADD) to compute the corresponding input space line and sample. Note that this computation includes the DEM height interpolated in step 1 above, which is new for LDCM.
3. From the input space line and sample calculate the reference year, day, and seconds, satellite position and velocity, and spacecraft attitude (roll-pitch-yaw). Note that the inclusion of roll-pitch-yaw here is new for LDCM.
4. Calculate the transformation matrix from earth fixed to orbit oriented.
5. Calculate the line of sight.

Calculate Position (oli_calc_position)

This function finds the satellite position, velocity, attitude, and time using the forward model. This unit invokes oli_findtime to get the time, oli_findatt to get the attitude, and l8_movesat to compute position and velocity. These sub-algorithms are described in the OLI LOS Projection ADD.

Get Latitude/Longitude (getlatlong)

This function finds the latitude/longitude given the L1G line/sample.

1. Find first order rotation coefficients if there is a rotation.
2. Find output projection coordinate of pixel.
3. Call projtran routine (see the LOS Projection ADD for details) to convert projection X/Y coordinates to the corresponding latitude/longitude.

4. Access the DEM to interpolate the height at the L1G line/sample coordinates. Note that this is a departure from the ALIAS heritage approach and is a consequence of using terrain corrected mensuration images.
5. Convert the latitude, longitude, and height into Cartesian ECEF coordinates (x,y,z) as described below.

Geodetic to Cartesian (geo_geod2cart)
This function converts geodetic coordinates (lat, lon, height) into Cartesian coordinates (x, y, z) as described in the LOS Projection ADD and reiterated below. Input latitude and longitude are in radians, height, semi-major axis, and output Cartesian position vector are in meters; flattening is a dimensionless number.

$$b = a (1 - f)$$
$$e^2 = 1 - b^2 / a^2$$
$$N = a / (1 - e^2 \sin^2(\varphi))^{1/2}$$
$$X = (N + h) \cos(\varphi) \cos(\lambda)$$
$$Y = (N + h) \cos(\varphi) \sin(\lambda)$$
$$Z = (N (1-e^2) + h) \sin(\varphi)$$

where:

X, Y, Z	-	ECEF coordinates
φ, λ, h	-	Geodetic coordinates (lat ϕ, long λ, height h)
N	-	Ellipsoid radius of curvature in the prime vertical
f	-	Ellipsoid flattening (f = 1 - b/a)
e^2	-	Ellipsoid eccentricity squared
a, b	-	Ellipsoid semi-major and semi-minor axes

Calculate Line of Sight (calc_line_of_sight)
This function calculates the line-of-sight angles from the satellite position to the ground point from their position coordinates.

For x, y, and z, assign: ecf_look = pixpos - satpos.

Perform matrix multiplication to transform earth fixed look vector to orbit oriented look vector (see the Earth Fixed to Orbit Oriented sub-algorithm below for the construction of the Tecf2oo transformation matrix):

$$[\text{Tecf2oo}]_{3x3} [\text{ecf\_look}]_{3x1} = [\text{oo\_look}]_{3x1}$$

Compute the along- and across-track angles:

psi = arctan(oo_look[0] / oo_look[2])
delta = arctan(oo_look[1] / oo_look[2])

Calculate Correction (calc_correction)
This function solves for the attitude and/or ephemeris correction using the Ground Control Points.

1. Initialize the correction parameter structure.
2. Allocate memory for residuals structure.

3. Begin the outlier detection and rejection iteration loop.
4. Prepare the residual file to be written to.
5. Reset the GCP information to its original state.
6. Initialize weight factor for observation and a priori parameters.
7. Iterate the precision correction solution process.
   a) Initialize the normal equations.
   b) For each GCP compute the observables (α and β), relate them to the correction parameters, and then form the normal equation to accumulate.
   c) Accumulate the normal equations by adding up information from each GCP.
   d) compute diff_time = gcps[gcp_num].time - ref_time[2]
   e) Write the residual information for this iteration. This will be done for each iteration. The structure get_residuals must be filled before writing to this file. We store the RMS residuals for the first and last iteration as solution quality metrics.
   f) Solve the Normal equations. Solve for the corrections from the normal equation using the Weighted Least Square sub-algorithm.
   g) If the parameter flag is 4 (weight factor estimation option):
      1. Estimate the variance factor with Minimum Norm Quadratic Unbiased Estimate (MINQUE).
      2. If MINQUE solution is obtained, compute the residual square sum.
      3. If MINQUE solution failed Try Maximum Likelihood Estimate (MLHE) solution.
      4. If MLHE fails, the solution can not be obtained.
      5. Calculate the posteriori standard error.

      Else If the parameter flag is not 4 (no a priori weight factor estimation is used):
      1. Compute the residual square sum.
      2. Calculate the posteriori standard error.
   h) Update the total correction estimate.
   i) Update the observable and orbit state for each GCP.
   j) If the sum of the absolute values of the elements of the across-track and along-track solution vectors are greater than 1 and the number of iterations is less than max iterations, iterate again, otherwise end iteration.
8. Calculate the residual in alpha and beta for each GCP.
9. Check the residuals for new outliers, if any are found continue the outlier iteration loop from step 3 above.
10. Extract the final solution and update the correction parameters.
11. Free memory.

Write Residuals (write_residuals)
This function writes out the residual for along- and across-track components for each GCP to the residual file.

For each GCP:
Rescale residuals to meters.
Compute the projection space value of the residuals.
Copy the information to the residual structure.
Write out the residual information.

Get Ground (get_ground)
This function calculates the projection (x/y) residual values in meters from the earth orbit delta and psi residual values.

1. Calculate the Earth Centered Fixed (ECF) to Orbit Oriented (OO) transformation system and transpose the matrix to get the OO to ECF matrix.
2. Given the satellite position and correction terms, calculate a new look vector.
3. Transform the vector from OO to ECF.
4. Convert the ECF latitude and longitude to projection in meters.
5. Convert the true latitude and longitude to projection in meters.
6. Subtract the projection and assign to residual.

Earth-fixed to Orbit-oriented (xxx_earth2orbit)

This function generates the transformation matrix from the Earth-fixed cartesian system to the orbit-oriented cartesian system as described in the Ancillary Data Preprocessing ADD and reiterated below. Note that the ECEF velocity vector is really the ECI velocity vector rotated into the ECEF coordinate system (i.e., it is still an inertial velocity) and does not include the relative Earth rotation velocity. This is done so that the ECEF velocity vector remains parallel to the attitude control reference X axis, which is defined in ECI coordinates.

The relationship between the orbital and Earth Centered coordinate systems is based on the spacecraft's instantaneous ECEF position and velocity vectors. The rotation matrix to convert from orbital to ECEF can be constructed by forming the orbital coordinate system axes in ECEF coordinates:

$$\vec{n} = -\frac{\vec{p}}{\left\|\vec{p}\right\|}$$

$$\vec{h} = \frac{\left(\vec{n} \times \vec{v}\right)}{\left\|\vec{n} \times \vec{v}\right\|}$$

$$\vec{cv} = \vec{h} \times \vec{n}$$

$$\left[\text{ORB2EC}\right] = \left[\vec{cv} \quad \vec{h} \quad \vec{n}\right]$$

where:

p = spacecraft position vector in ECEF
v = spacecraft velocity vector in ECEF
n = nadir vector direction
h = negative of angular momentum vector direction
cv = circular velocity vector direction
[ORB2ECEF] = rotation matrix from orbital to ECEF

The transformation from orbital to ECEF coordinates is the inverse of the ECEF to orbital transformation matrix. Since the ECEF to orbital matrix is orthogonal the inverse is also equal to the transpose of the matrix.

$$[\text{ORB2ECEF}] = [\text{ECEF2ORB}]^{-1} = [\text{ECEF2ORB}]^{T}$$

Detect Outliers (det_outliers)
This function detects GCP outliers using the residuals and normal equations. Given a tolerance value, outliers are removed within the data set until all values deemed as "non-outliers" or "valid" fall inside the confidence interval of a T-distribution. The tolerance, or associated confidence interval, is specified per run and usually lies between 0.9-0.99. The default value is 0.95. The number of degrees of freedom of the data set is equal to the number of valid data points minus one. The steps involved in this outlier procedure are as follows:

1. Calculate standard deviation of all valid points in the data set.
2. Loop on "valid" data points until no outliers are found.
   a) Find two tailed T-distribution (*T*) value for current degree of freedom and confidence level specified α.
   b) Calculate largest deviation allowable for the specified degree of freedom and α. This is not scaled by σ since the residuals themselves are normalized by σ in step c below.
      $\Delta = T$
   c) For each data point, compute the along- and across-track weight factors using equation (7.18) above and the normalized and weighted along- and across-track residuals using equation (7.27) above.
   d) Find the data point with the largest normalized and weighted residual.
   e) If maximum residual value found in step d is less than Δ, then exit
   f) If value found in step d is greater than Δ, then flag the data point as an outlier and calculate the standard deviation of the new set of "valid" data points.

Get Correction (get_correction)
This function extracts the estimated correction parameters and their covariance matrix from the Weighted Least Square solution, update the correction parameter structure.

1. Record the reference time for the correction.
2. Extract satellite position corrections
3. Extract satellite velocity corrections
4. Extract satellite attitude angle corrections
5. Extract satellite attitude angle rate corrections
6. Record the covariance matrix for the correction parameters

Reset Observations (resetobserv)
This function resets the satellite state vector and the look angles corresponding to each GCP to their original values. This resets the inputs for the next iteration of the outlier loop.

Initialize Precision (initial_precision)
This function initializes the normal matrix for attitude and ephemeris correction estimate by least-square solutions.

1. Initialize the observational and a priori part of the normal equation, obs_mx, obs_rgt, apr_corr, apr_wgt_par, to zero or almost zero.
2. if param_flag = both or input weights are provided, estimate all corrections.
3. Form the a priori normal matrix for the parameters.
4. Form the a priori right-side term for the parameters.

5. Subtract the current net correction (Yb) terms from the right hand side to restrain the magnitude of the net correction.
6. if param_flag = eph_yaw, estimate orbit corrections:
   a) set zero a priori mean for roll
   b) set zero a priori mean for roll dot
   c) set huge a priori weight for roll
   d) set huge a priori weight for roll dot
   e) set zero a priori mean for pitch
   f) set zero a priori mean for pitch dot
   g) set huge a priori weight for pitch
   h) set huge a priori weight for pitch dot
7. if param_flag = att_orb, estimate attitude corrections:
   a) set zero a priori mean for dy
   b) set zero a priori mean for dy dot
   c) set huge a priori weight for dy
   d) set huge a priori weight for dy dot
   e) set zero a priori mean for dx
   f) set zero a priori mean for dx dot
   g) set huge a priori weight for dx
   h) set huge a priori weight for dx dot
8. if time_flag = FALSE
   a) Block out the rate terms by setting a huge weight for zero apriori mean.
9. Initialize the number of observations.
10. Initialize weighted residual square summation.

Process One GCP (process_one_gcp)

This function updates the normal equation of the least-square problem for correction solution by adding one Ground Control Point.

Calculate the transformation matrix from ECF to Orbit system
Calculate the line-of-sight angles for GCP

Note: The look vectors here should be in the Orbit reference system.

If the line of sight angle for the pixel P_i is from the forward model (in the spacecraft-fixed system), then it should be transformed into the Orbit reference system (through matrix A(roll, pitch, yaw)) first before the observable alpha and beta can be formed.

Compute the observable alpha and beta
If not an outlier:
Relate the observable to correction parameters
Update the weighted square summation of observation
Accumulate the normal equation contribution for alpha
Accumulate the normal equation contribution for beta

Partial (partial)

This function composes the partial coefficients matrix of the observation equation, given the angle delta for one GCP.

Partial Attitude (partial_att)

This function composes the partial coefficients matrix of the observation equation for param_flag = "att_orb", estimating attitude plus height corrections.

Calculate the constants needed for the partial derivative (H) matrix calculation such as sin(delta), cos(delta), and satellite radius.
The side perpendicular to the look vector = satellite_radius * sindelta
Compose the H matrix by finding:

- alpha w.r.t roll, microradian
- alpha w.r.t pitch, microradian
- alpha w.r.t yaw, microradian
- alpha w.r.t. dz, meter scaled to microradian
- beta w.r.t roll, microradian
- beta w.r.t. pitch, microradian
- beta w.r.t. yaw, microradian

Partial Ephemeris (partial_eph)

This function composes the partial coefficients matrix of the observation equation for param_flag = "eph_yaw", estimating ephemeris plus yaw corrections.

Calculate the constants needed for H calculation by assigning sin(delta), cos(delta) and satellite radius.
Compose the H matrix by finding:

- alpha w.r.t. dy, meter scaled to microradian
- alpha w.r.t. dz, meter scaled to microradian
- alpha w.r.t. yaw, microradian
- beta w.r.t. dx, meter scaled to microradian
- beta w.r.t. yaw, microradian

Partial All (partial_all)

This function composes the partial coefficients matrix of the observation equation for param_flag = "both" or "weight", estimating both attitude and ephemeris corrections. Note that this is the normal case.

Calculate the constants needed for H calculation sin(delta), cos(delta), and satellite radius (see equations (3.6) and (3.7) above).
Compose the H matrix:

- alpha w.r.t. roll, microradian
- alpha w.r.t. pitch, microradian
- alpha w.r.t. yaw, microradian
- alpha w.r.t. dy, meter scaled to microradian
- alpha w.r.t. dz, meter scaled to microradian
- beta w.r.t. dx, meter scaled to microradian
- beta w.r.t. roll, microradian
- beta w.r.t. pitch, microradian
- beta w.r.t. yaw, microradian

Accumulate Normal Equation (accum_normal_equation)

This function accumulates the normal equation of the least-square problem by adding one observation.

Update the n x n normal matrix by accumulating:
H_transpose * wo * H

Update the n x 1 right-hand-side array of the normal equation by adding:
H_transpose * wo * obs

where:
H is the matrix of partial derivatives
wo is the observation weight
obs is the observation value

Weighted Least Square (weighted_least_square)

This function solves the weighted least square problem with nxn normal matrix.

Form the normal equation for the Weighted Least Square (WLS) problem, including any weight factors:
A[i][j] = weight_factor_for_observation * normal_matrix_for_observation[i][j]
Augment the diagonal terms using the apriori observations:
A[i][i] += weight_factor_for_apriori * normal_matrix_for_apriori[i]

Form the constant vector including both observations and apriori contributions:
L[i] = weight_factor_for_observation * observation_rhs[i]
+ weight_factor_for_apriori * apr_corr[i]

Solve the equation:
solution = sol_Ya = A^{-1} L

Note that the inverted normal equation matrix (A^{-1}) is returned along with the solution so that it can be used to construct the solution aposteriori covariance matrix.

MINQUE (minque)

This function estimates the variance factor with MINQUE (Minimum Norm Quadratic Unbiased Estimate).

let:
wght_rss_obs = weighted residual square for observation
cov_mx = Inverse of the WLS problem normal matrix
obs_mx = the observation part of the normal matrix
apr_wgt_par = the a priori weights loaded into a diagonal weight matrix
wgt_fact_obs = the estimated variance factor for the observation
wgt_fact_apr = the estimated variance factor for the a priori variance

compute the weighted residual square for the observation (rss_obs)

compute the weighted residual square for the a priori parameters (rss_apr)

Allocate memory for arrays

compute the trace coefficients for the weight estimate equation

cc2 = cov_mx * apr_wgt_par
cc1 = cov_mx * obs_mx
s1 = ngcp - 2tr[cc1] + tr[cc1 * cc1] ref. equation (5.3)
s2 = n_aprior - 2tr[cc2] + tr[cc2 * cc2] ref. equation (5.4)
s12 = tr[cc1 * cc2] ref. equation (5.5)

solve for the weight factors:

ss1 = s1 * s2 - s12 * s12
wgt_fact_obs = (rss_obs * s2 - rss_apr * s12) / ss1
wgt_fact_apr = (rss_apr * s1 - rss_obs * s12) / ss1

If wgt_fact_obs and wgt_fact_apr are less than 0.0--return, minque failed.
Run WLS where the scale factor for the weight of observation and a priori are 1/wgt_fact_obs and 1/wgt_fact_apr respectively.
If WLS fails, return minque with failed status.
If WLS returns a non-error value, assign wght_rss_obs.

Residual Square Sum (resquare)

This function computes the residual square sum by adding the dot product of:

$sol\_Ya^T * obs\_mx * sol\_Ya - 2 * obs\_rgt^T * sol\_Ya$

where:

sol_Ya is the weighted least squares solution vector
obs_mx is the normal equation matrix
obs_rgt is the right hand side vector of the normal equations

to the observation square sum (post_sig).

MLHE (mlhe)

This function estimates the variance factor with MLHE (Maximum Likelihood Estimate).

Initialize the weight factor to zero.
Iterate the estimation of the weight factors.
Compute the weighted residual square for the observation and for the apriori parameters.
Compute the weight factor estimate.
Compute the weight factor difference for this iteration.
Solve the new WLS solution with the new weight factors.
Compute the final variance factor estimate.

New Observation Angle (new_observ_angle)

This function updates the satellite state vector and the look angles corresponding to each GCP, according to the correction parameters, for the purpose of iteration.

Extract the orbit and attitude correction parameters from the solution vectors.

Orbit corrections:

dorbit[0] = sol_Ya[3]
dorbit[1] = sol_Ya[4]

dorbit[2] = sol_Ya[5]
orbit_rate[0] = sol_Ya[9]
orbit_rate[1] = sol_Ya[10]
orbit_rate[2] = sol_Ya[11]

Attitude corrections:
datt[0] = sol_Ya[0]
datt[1] = sol_Ya[1]
datt[2] = sol_Ya[2]
att_rate[0] = sol_Ya[6]
att_rate[1] = sol_Ya[7]
att_rate[2] = sol_Ya[8]

For each GCP
Calculate the orbit perturbation and update the orbit state vector
Calculate the attitude perturbation and update the look angles

Update Ephemeris (update_eph)
This function calculates the orbit position change and updates the ephemeris data in Earth Fixed system.

Construct the ECF to orbital transformation T_{ef2oo} from the input position and velocity vectors (using xxx_earth2orbit).

Take the transpose of (the orthogonal matrix) T_{ef2oo} to find the inverse T_{oo2ef}.

Transform the input orbital position and velocity corrections to ECF using T_{oo2ef}.

Update the input ECF position and velocity by adding the transformed position and velocity corrections.

Calculate New Look Angles (newlook)
This function calculates the new look angles by adding the attitude angle perturbation. The heritage ALIAS implementation was modified as described below to account for applying the attitude corrections in the ACS rather than the orbital coordinate system.

Convert the units of the attitude corrections (to radians).
Construct the look vector from the two look angles.

look_vector[0] = tan(psi)
look_vector[1] = tan(delta)
look_vector[2] = 1.0

Convert the orbital look vector to the ACS coordinate system:
Use the roll-pitch-yaw values for this GCP to construct the orbital to ACS rotation matrix $\mathbf{M}_{ORB2ACS} = [\mathbf{M}_{ACS2ORB}]^T$.
Where: $\mathbf{M}_{ACS2ORB}$ =

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

Convert look_vector to ACS_look_vector by multiplying it by $[\mathbf{M}_{ACS2ORB}]^T$:

ACS_look_vector = $[\mathbf{M}_{ACS2ORB}]^T$ look_vector

Use the attitude corrections to construct the ACS correction rotation matrix $\mathbf{M}_{Precision}$:

8. Compute the precision correction at the time (t_att = att_seconds + att_time) corresponding to the attitude sample:
   a. roll_corr = roll_bias + roll_rate * (t_att – t_ref – image_seconds)
   b. pitch_corr = pitch_bias + pitch_rate * (t_att – t_ref – image_seconds)
   c. yaw_corr = yaw_bias + yaw_rate * (t_att – t_ref – image_seconds)

   Note that only the seconds of day fields are needed for the attitude and image epochs as they are constrained to be based on the same year and day.
9. Compute the rotation matrix corresponding to roll_corr, pitch_corr, and yaw_corr ($\mathbf{M}_{Precision}$) using the same equations used for $\mathbf{M}_{ACS2ORB}$ above.

Apply the attitude corrections to the look vector by multiplying by $\mathbf{M}_{Precision}$:

ACS_pert_look_vector = $\mathbf{M}_{Precision}$ ACS_look_vector

Rotate line of sight back to the orbital coordinate system using the transpose of the $\mathbf{M}_{ORB2ACS}$ matrix, which is the same as $\mathbf{M}_{ACS2ORB}$:

pert_look_vector = $\mathbf{M}_{ACS2ORB}$ ACS_pert_look_vector

Note that this can be achieved with a single rotation of:

$\mathbf{M}_{corr} = \mathbf{M}_{ACS2ORB}\ \mathbf{M}_{Precision}\ [\mathbf{M}_{ACS2ORB}]^T$

pert_look_vector = $\mathbf{M}_{corr}$ look_vector

Calculate the new look angles:

psi = arctan(pert_look_vector[0]/pert_look_vector[2])

delta = arctan(pert_look_vector[1]/pert_look_vector[2])

Calculate Observation Residual (observation_residual)

This function corrects the final values of alpha and beta for all GCPs for the effects of the final solution iteration. These values are updated by process_one_gcp for all but the final iteration.

For each GCP

Calculate the full partial coefficients matrix for alpha and beta.

For all 6 elements

Calculate the residual for alpha by subtracting the calculated observation.

gcps.va = gcps.va - H1 *Ya

Calculate the residual for beta by substracting the calculated observation.

gcps.vb = gcps.vb - H2 *Ya

Where Ya is the vector containing the incremental parameter corrections for the last iteration.

Finish Processing (finish_processing)
This function updates the OLI model file and writes to the solution and residual files. It has new functions added to check the solution quality statistics (pre-fit RMS, post-fit RMS, outlier percent, number of valid GCPs) to determine if the solution was successful.

Compute the percentage of GCPs that were declared outliers:
  percent_outlier = num_outlier / num_GCP * 100
Compute the number of valid GCPs:
  num_valid = num_GCP – num_outlier
Check the pre-fit RMS, post-fit RMS, percent_outlier, and num_valid metrics against the thresholds (maximum pre-fit RMS, maximum post-fit RMS, maximum outlier percentage, minimum number of valid GCPs) from the CPF.
If the pre- and post-fit RMS values are both below the thresholds, and either the percent_outlier metric is below threshold or the num_valid metric is above threshold:
    Update the model to make a precision model.
    Fill the gcp_solution structure with the appropriate values.
    Write to the solution file.
    Return success status.
Else return failure status.

Update LOS Model (oli_update_model)
This function updates the LOS model file with the precision correction values. The LOS model will be read from the LOS model file, the new precision correction values will be placed in the LOS model structure, the LOS model will be processed with the new precision correction values, and the new precision LOS model structure will be output to the precision LOS model file.

Unlike the heritage ALIAS approach, not only are the precision correction parameters stored in the LOS model, they are also applied to both the ephemeris and attitude data sequences. This is captured in the LOS model by storing both original and corrected attitude and ephemeris data sequence. This update procedure operates as follows:

Correct Attitude Sub-Algorithm (l8_correct_attitude)
This function applies the ACS/body space attitude corrections computed by the LOS/precision correction procedure to the attitude data sequence. It outputs a parallel table of roll-pitch-yaw values with the precision corrections applied. This "corrected" table is created by the LOS Model Creation algorithm but initially it is identical to the original attitude data sequence.

The sequence of transformations required to convert a line-of-sight in the OLI instrument coordinate system, generated using the Legendre polynomials, is:

$$\underline{\mathbf{x}}_{ECEF} = \mathbf{M}_{ORB2ECEF}\,\mathbf{M}_{ACS2ORB}\,\mathbf{M}_{Precision}\,\mathbf{M}_{OLI2ACS}\,\underline{\mathbf{x}}_{OLI}$$

where: $\underline{\mathbf{x}}_{OLI}$ is the Legendre-derived instrument LOS vector
$\mathbf{M}_{OLI2ACS}$ is the OLI to ACS alignment matrix from the CPF
$\mathbf{M}_{Precision}$ is the correction to the attitude data computed by the LOS/precision correction procedure
$\mathbf{M}_{ACS2ORB}$ is the spacecraft attitude (roll-pitch-yaw)
$\mathbf{M}_{ORB2ECEF}$ is the orbital to ECEF transformation computed using the ECEF ephemeris

$\underline{\mathbf{x}}_{ECEF}$ is the LOS vector in ECEF coordinates

Note that in the heritage ALIAS implementation the sequence was:

$$\underline{\mathbf{x}}_{ECEF} = \mathbf{M}_{ORB2ECEF}\,\mathbf{M}_{Precision}\,\mathbf{M}_{ACS2ORB}\,\mathbf{M}_{OLI2ACS}\,\underline{\mathbf{x}}_{OLI}$$

For nadir-viewing imagery the $\mathbf{M}_{ACS2ORB}$ matrix is nearly identity, so there is little difference. Since OLI will occasionally be viewing off-nadir and it is more natural to model attitude errors in the ACS/body coordinate system, the order has been reversed for LDCM. The impact is minimal in the model and LOS projection but becomes more important for the LOS/precision correction algorithm.

This new sub-algorithm pre-computes the $\mathbf{M}_{ACS2ORB}\,\mathbf{M}_{Precision}$ combination and stores the corresponding corrected roll-pitch-yaw attitude sequence in the model structure. This approach has several advantages:

4. It streamlines the application of the model for LOS projection by removing the step of explicitly applying the precision correction.
5. It allows for the use of a more complex correction model in the future since the application of the model is limited to this unit. Note that the Earth-view attitude correction model consists of the following model parameters:
   Precision reference time: t_ref in seconds from the image epoch (nominally near the center of the image time window)
   Roll bias and rate corrections: roll_bias, roll_rate
   Pitch bias and rate corrections: pitch_bias, pitch_rate
   Yaw bias and rate corrections: yaw_bias, yaw_rate
   This model is dealt with in more detail in the line-of-sight correction algorithm description.
6. Retaining both the original and corrected attitude sequences in the model make the model self-contained and will make it unnecessary for the LOS/precision correction algorithm to access the preprocessed ancillary data.

The disadvantage is that it doubles the size of the attitude data in the model structure.

The construction of the corrected attitude sequence proceeds as follows:

For each point in the attitude sequence j = 0 to K-1:

1. Compute the rotation matrix corresponding to the j^{th} roll-pitch-yaw values:

$\mathbf{M}_{ACS2ORB}$ =

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

2. Compute the precision correction at the time (t_att = att_seconds + att_time(j)) corresponding to the attitude sample:
   a. roll_corr = roll_bias + roll_rate * (t_att – t_ref – image_seconds)
   b. pitch_corr = pitch_bias + pitch_rate * (t_att – t_ref – image_seconds)
   c. yaw_corr = yaw_bias + yaw_rate * (t_att – t_ref – image_seconds)

   Note that only the seconds of day fields are needed for the attitude and image epochs as they are constrained to be based on the same year and day.
3. Compute the rotation matrix corresponding to roll_corr, pitch_corr, and yaw_corr ($\mathbf{M}_{Precision}$) using the same equations presented in step 1 above.
4. Compute the composite rotation matrix: $\mathbf{M} = \mathbf{M}_{ACS2ORB}\,\mathbf{M}_{Precision}$

5. Compute the composite roll-pitch-yaw values:

$$\text{roll}' = -\tan^{-1}\left(\frac{M_{2,1}}{M_{2,2}}\right)$$

$$\text{pitch}' = \sin^{-1}\left(M_{2,0}\right)$$

$$\text{yaw}' = -\tan^{-1}\left(\frac{M_{1,0}}{M_{0,0}}\right)$$

6. Store the composite roll'-pitch'-yaw' values in the j^{th} row of the corrected attitude data table.

Correct Ephemeris Sub-Algorithm (l8_convert_ephem)

The heritage ALIAS function converted the ephemeris information (position and velocity) from the Earth Centered Inertial (ECI J2000) system to the Earth Centered Earth Fixed (ECEF) system and applied the ephemeris corrections computed in the LOS/precision correction procedure to both ephemeris sets. Since both ECI and ECEF representations of the ephemeris are now provided by the ancillary data preprocessing algorithm, the first portion of the heritage algorithm is no longer necessary.

The precision correction parameters are stored in the LOS model in the spacecraft orbital coordinate system as three position (x_bias, y_bias, z_bias) corrections and three velocity (x_rate, y_rate, z_rate) corrections that, like the attitude corrections, are relative to t_ref. These values must be converted to the ECEF and ECI coordinate systems. Once the precision correction is determined in the ECEF/ECI coordinate system, the ECEF/ECI ephemeris values can be updated with the precision parameters.

Loop on LOS model ephemeris points j = 0 to N-1

Compute the precision correction:

Calculate delta time for precision correction:

dtime = ephem_seconds + ephem_time(j) – t_ref – image_seconds

Calculate the change in X, Y, Z due to precision correction. Corrections are in terms of spacecraft orbital coordinates.

dx orb = model precision x_bias + model precision x_rate * dtime
dy orb = model precision y_bias + model precision y_rate * dtime
dz orb = model precision z_bias + model precision z_rate * dtime

where:
model precision x_bias = precision (orbital coord sys) update to X position
model precision y_bias = precision (orbital coord sys) update to Y position
model precision z_bias = precision (orbital coord sys) update to Z position
model precision x_rate = precision (orbital coord sys) update to X velocity
model precision y_rate = precision (orbital coord sys) update to Y velocity
model precision z_rate = precision (orbital coord sys) update to Z velocity

Construct precision position and velocity "delta" vectors.

$$[\text{dorb}] = \begin{bmatrix} \text{dx orb} \\ \text{dy orb} \\ \text{dz orb} \end{bmatrix}$$

$$[\text{dvorb}] = \begin{bmatrix} \text{model precision x rate} \\ \text{model precision y rate} \\ \text{model precision z rate} \end{bmatrix}$$

Calculate the orbit to ECF transformation [ORB2ECEF] using ECEF ephemeris (See the ancillary data preprocessing ADD for this procedure).

Transform precision "delta" vectors to ECEF.

$$[\text{def}] = [\text{ORB2ECEF}][\text{dorb}]$$

$$[\text{dvef}] = [\text{ORB2ECEF}][\text{dvorb}]$$

Adjust ECEF ephemeris by the appropriate "delta" precision vector and store the new ephemeris in the model. These ephemeris points will be used when transforming an input line/sample to an output projection line/sample.

$$\text{model ef postion} = \text{ephemeris ecef postion} + \text{decf}$$

$$\text{model ef velocity} = \text{ephemeris ecef velocity} + \text{dvecf}$$

where:

All parameters are 3x1 vectors
ephemeris ecef values are the interpolated one-second ephemeris values in ECEF coordinates

Calculate the orbit to ECI transformation [ORB2ECI] using ECI ephemeris.

Transform precision "delta" vectors to ECI.

$$[\text{deci}] = [\text{ORB2ECI}][\text{dorb}]$$

$$[\text{dveci}] = [\text{ORB2ECI}][\text{dvorb}]$$

Adjust ECI ephemeris by the appropriate "delta" precision vector and store the new ephemeris in the model. These ephemeris points will be used with lunar/stellar observations.

$$\text{model eci postion} = \text{ephemeris eci postion} + \text{deci}$$

$$\text{model eci velocity} = \text{ephemeris eci velocity} + \text{dveci}$$

where:

All parameters are 3x1 vectors

ephemeris eci values are the interpolated one-second ECI ephemeris

Convert the Net Attitude Corrections to Alignment Angles (calc_alignment)
This new sub-algorithm combines the newly computed attitude correction with the OLI sensor alignment matrix from the LOS model to construct corrected alignment angles.

Compute the precision correction at the reference time t_ref:
roll_corr = roll_bias
pitch_corr = pitch_bias
yaw_corr = yaw_bias

Compute the rotation matrix corresponding to roll_corr, pitch_corr, and yaw_corr ($\mathbf{M}_{Precision}$) using the standard rotation matrix equations:

$\mathbf{M}_{Precision}$ =

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

Extract the ACS to OLI alignment matrix, $\mathbf{M}_{ACS2OLI}$, from the OLI LOS model, and take the transpose to compute $\mathbf{M}_{OLI2ACS}$.

Compute the composite alignment matrix: $\mathbf{M} = \mathbf{M}_{Precision}\,\mathbf{M}_{OLI2ACS}$

Compute the composite roll-pitch-yaw alignment angles:

$$roll' = -\tan^{-1}\left(\frac{M_{2,1}}{M_{2,2}}\right)$$

$$pitch' = \sin^{-1}\left(M_{2,0}\right)$$

$$yaw' = -\tan^{-1}\left(\frac{M_{1,0}}{M_{0,0}}\right)$$

Extract the orbital ephemeris biases from the precision solution: (x_bias, y_bias, z_bias).

Extract the attitude bias correction and ephemeris bias correction covariance terms from the precision solution covariance matrix:

The solution provides:

$$X = \begin{bmatrix} droll \\ dpitch \\ dyaw \\ dX \\ dY \\ dZ \\ droll/dt \\ dpitch/dt \\ dyaw/dt \\ dX/dt \\ dY/dt \\ dZ/dt \end{bmatrix}$$ with covariance Cov

We want to form:

$$X = \begin{bmatrix} droll \\ dpitch \\ dyaw \\ dX \\ dY \\ dZ \end{bmatrix}$$

With (see note #2): $$CovX = \begin{bmatrix} Cov[0][0] & Cov[0][1] & Cov[0][2] & Cov[0][3] & Cov[0][4] & Cov[0][5] \\ Cov[1][0] & Cov[1][1] & Cov[1][2] & Cov[1][3] & Cov[1][4] & Cov[1][5] \\ Cov[2][0] & Cov[2][1] & Cov[2][2] & Cov[2][3] & Cov[2][4] & Cov[2][5] \\ Cov[3][0] & Cov[3][1] & Cov[3][2] & Cov[3][3] & Cov[3][4] & Cov[3][5] \\ Cov[4][0] & Cov[4][1] & Cov[4][2] & Cov[4][3] & Cov[4][4] & Cov[4][5] \\ Cov[5][0] & Cov[5][1] & Cov[5][2] & Cov[5][3] & Cov[5][4] & Cov[5][5] \end{bmatrix}$$

The covariance matrix captures the correlations between the attitude and ephemeris correction parameters (e.g., roll-Y and pitch-X).

The following fields are output to the alignment characterization database:

- Reference time: image epoch year, image epoch day, image epoch second + t_ref
- Alignment vector: X above
- Alignment covariance: CovX above
- RMS GCP fit
- Number of GCPs used
- Outlier threshold used
- Scene off-nadir roll angle
- Control type flag (DOQ or GLS)
- L0Rp ID

Work Order ID
WRS Path/Row

LOS Model Correction Output Summary

The primary output of the LOS model correction algorithm is the updated "precision" LOS model. This model has the same structure as the input LOS model which is described in the LOS Model Creation ADD. Though the model structure is the same, the corrected ECI position and velocity and the corrected ECEF position and velocity sections of the Ephemeris Model, the corrected roll-pitch-yaw section of the Attitude Model, and the Precision Correction Model all contain updated values as a result of the LOS model correction algorithm.

The contents of the output LOS Model Correction Solution File are presented in Table 1 below. This report file documents the results of the LOS model correction procedure. It contains standard header fields common to all geometric report files.

Field	Description
1. Date and time	2. Date (day of week, month, day of month, year) and time of file creation.
3. Spacecraft and instrument source	4. LDCM and OLI
5. Processing Center	6. EROS Data Center SVT
7. Work order ID	8. Work order ID associated with processing (blank if not applicable)
9. WRS path/row	10. WRS path and row
11. Software version	12. Software version used to create report
13. Off-nadir angle	14. Off-nadir roll angle of processed image file
15. Acquisition Type	16. Earth viewing or Lunar
17. L0Rp ID	18. Input L0Rp image ID
19. L1G image file	20. Name of L1G used to measure GCPs
21. Precision solution reference time	22. Time reference for model correction parameters as year, day of year and seconds of day.
23. Roll-pitch-yaw attitude corrections	24. Attitude bias corrections in microradians
25. Roll-pitch-yaw rate corrections	26. Attitude rate corrections in microradians/second
27. Roll-pitch-yaw standard deviations	28. Attitude bias parameter sigmas in microradians
29. Roll-pitch-yaw rate std. devs.	30. Attitude rate parameter sigmas in microrads/sec
31. Ephemeris position corrections	32. Ephemeris X-Y-Z bias corrections in meters
33. Ephemeris velocity corrections	34. Ephemeris Vx-Vy-Vz corrections in meters/second
35. Position standard deviations	36. Ephemeris X-Y-Z sigmas in meters
37. Velocity standard	38. Ephemeris Vx-Vy-Vz sigmas in

deviations	meters/second
39. Across-track covariance matrix	40. 6-by-6 covariance matrix for roll, Y, Z, roll rate, Vy, Vz correction parameters.
41. Along-track covariance matrix	42. 6-by-6 covariance matrix for X, pitch, yaw, Vx, pitch rate, yaw rate correction parameters.
43. Spacecraft roll-pitch-yaw at solution reference time	44. Spacecraft attitude at solution reference time in microradians

Table 1: LOS Model Correction Solution Output File Contents

The contents of the LOS Model Correction Residuals file are shown in Table 2 below. This file documents the GCP residuals for the final set of GCPs (after the outlier rejection loop has found no additional outliers), including the residuals for each iteration of the weighted least squares solution procedure. It thus contains both the initial (pre-correction) and final (post-correction) residuals. This file is used as an input by the Geodetic Accuracy Assessment algorithm. The output residual file also contains the standard report header mentioned above.

Field	**Description**
45. Date and time	46. Date (day of week, month, day of month, year) and time of file creation.
47. Spacecraft and instrument source	48. LDCM and OLI
49. Processing Center	50. EROS Data Center SVT
51. Work order ID	52. Work order ID associated with processing (blank if not applicable)
53. WRS path/row	54. WRS path and row
55. Software version	56. Software version used to create report
57. Off-nadir angle	58. Off-nadir roll angle of processed image file
59. Acquisition Type	60. Earth viewing or Lunar
61. L0Rp ID	62. Input L0Rp image ID
63. L1G image file	64. Name of L1G used to measure GCPs
65. Number of GCPs used	66. Number of valid (non-outlier) GCPs
67. Heading for individual GCPs	68. One line of ASCII text containing column headings for the individual GCP fields.
For each iteration:	
Iteration number	Starts with 0 for initial (uncorrected) residuals and ends with "Final" for results of last iteration.
For each GCP:	
Point ID	GCP ID (see GCP Correlation ADD for details)
Predicted L1G Line	Predicted L1G line location
Predicted L1G Sample	Predicted L1G sample location
GCP Time of Observation	Seconds from image epoch time
Latitude	GCP latitude in degrees
Longitude	GCP longitude in degrees
Height	GCP height in meters
Across-track Angle (delta)	Across-track LOS angle in degrees
Across-track Residual	Residual on delta converted to meters

Along-track Residual	Residual on psi converted to meters
Y Residual	Residual in Y/line direction in meters
X Residual	Residual in X/sample direction in meters
Outlier Flag	0 for outlier, 1 for valid GCP
GCP Source	DOQ or GLS

Table 2: LOS Model Correction Residuals Output File Contents

The fields stored in the characterization database for future sensor alignment calibration operations are listed in Table 3 below.

Field	**Description**
69. Work order ID	70. Work order ID associated with processing
71. WRS path/row	72. WRS path and row
73. L0Rp ID	74. Input L0Rp image ID
75. Control Type	76. DOQ or GLS
77. Off-nadir angle	78. Off-nadir roll angle of scene (in degrees)
79. Number of GCPs used	80. Number of valid (non-outlier) GCPs
81. Outlier threshold used	82. Confidence level used for outlier rejection threshold
83. RMS GCP Fit	84. RMS of final iteration across- and along-track residuals in meters. This field would subsequently be used to identify entries that may be suspect due to poor fits to the ground control.
85. Precision solution reference time	86. Time reference for model correction parameters as year, day of year and seconds of day.
87. Roll-pitch-yaw alignment angles	88. Composite alignment angles in microradians
89. Ephemeris position corrections	90. Ephemeris X-Y-Z bias corrections in meters
91. Alignment covariance matrix	92. 6-by-6 covariance matrix for roll, pitch, yaw, X, Y, Z correction parameters.

Table 3: Model/Alignment Characterization Database Output Fields

The fields stored in the characterization database to support future GCP quality assessment and improvement activities are listed in Table 4 below (see note #3). Only the residuals for non-outlier GCPs from the initial (zeroth) iteration are written to the characterization database.

Field	**Description**
For each GCP:	
93. Work order ID	94. Work order ID associated with processing
95. WRS path/row	96. WRS path and row
97. L0Rp ID	98. Input L0Rp image ID
Point ID	GCP ID (see GCP Correlation ADD for details)
GCP Time of Observation	Year, day of year, and seconds of day
Ephemeris Position	Spacecraft ECEF position at GCP time (meters)

Ephemeris Velocity	Spacecraft ECEF velocity at GCP time (meters/sec)
Spacecraft Roll-Pitch-Yaw	Spacecraft roll-pitch-yaw at GCP time (radians)
True Latitude	GCP latitude in radians
True Longitude	GCP longitude in radians
True Height	GCP height in meters
Apparent Latitude	Latitude measured in L1G image in radians
Apparent Longitude	Longitude measured in L1G image in radians
Apparent Height	Height interpolated from DEM in meters
GCP Source	DOQ or GLS

Table 4: GCP Residual Characterization Database Output Fields

7.2.3.8 Maturity

Though much of the ALI model correction algorithm was reusable there were several areas where changes were required:

9. Logic which computes the OLI sensor alignment corrections implied by the precision attitude and ephemeris corrections has been added to this algorithm (it runs as a pre-process in the ALIAS alignment calibration algorithm) to ensure that the computed corrections are applied to the proper sensor alignment matrix. Storing only the corrections leaves open the question of what alignment they are relative to. This was not a problem in the heritage systems (L7 IAS, ALIAS) because the alignment calibration process was run as one continuous flow using the same set of data throughout. This approach limited the number of scenes that could be processed and restricted the order of processing to be in data acquisition order. This restriction will be lifted for OLIAS so that a much larger volume of data can be reduced and trended for subsequent offline analysis. This requires the trended data to be converted to apparent alignment angles so that acquisitions processed using different alignment calibrations can be compared.
10. The covariance data that are trended for subsequent use in alignment calibration are a subset of the full precision solution covariance.
11. Trending of a slightly enhanced version of the initial (zeroth) iteration GCP residuals has been added to support offline research into large scale area triangulation. The path/row, date/time, GCP ID, true position, and apparent (mensuration image) position are recorded for all non-outlier GCPs.
12. Since the precision correction process will likely be run prior to any cloud screening and will therefore frequently fail due to the inability to correlate GCPs in cloud covered imagery, thresholds and bounds will need to be developed to detect cases in which the solution has failed. In this case, scene processing would fail over to a terrain-corrected systematic data flow. The prototype computes and reports three quality metrics: prefit GCP RMS, postfit GCP RMS, and percent GCP outliers, but does not apply any threshold logic. The operational version should apply thresholds on the pre-fit and post-fit GCP RMSE values, and make sure that either a sufficient number of valid GCPs were used or that the percentage of GCPs declared outliers was not too high.
13. Using a systematic terrain corrected image for GCP mensuration instead of the heritage systematic image required some modifications to the GCP processing logic. Specifically, the DEM elevation associated with the measured GCP image position is used to construct the “apparent” LOS instead of using zero for a LOS projected to the ellipsoid surface as the heritage algorithm does. Note that, while the actual GCP elevation could be used, this would introduce error that would grow with the misregistration between the systematic image and the

DEM, making the simpler approach less robust. This change was motivated by the large GCP search areas that would be required in systematically corrected images for off-nadir scenes in high elevation areas.

14. Using the DEM as the source of “apparent” GCP height allows the algorithm to support either terrain corrected or systematic image inputs. If an input DEM is not provided, the “apparent” GCP height will be set to zero as it is now. If an input DEM is provided, it will be used as the source of the “apparent” GCP height. Note that the capability to use SCA-combined mensuration images only applies for terrain corrected images.
15. The heritage ALIAS implementation generates a fatal error if the square root of a negative number is encountered while computing partial derivatives. This can happen in the case of an invalid GCP measurement. This will be enhanced for OLI to adopt the logic used in Landsat 7 wherein this condition is detected and used to declare the offending point an outlier rather than generating a terminal error condition.

7.2.3.9 Notes

Some additional background assumptions and notes include:

6. The heritage aliprecision process uses the DDR for the L1G mensuration image to retrieve the image framing and projection parameter information necessary to convert output space line/sample coordinates to latitude/longitude but the same information is available in the grid file, so either could be used.
7. The extent to which the model creation logic must be rerun was scaled back as compared to the heritage implementation. The precision ephemeris corrections are embedded in the model ephemeris so it must be regenerated but full model reprocessing is not truly necessary. This was done in the past for convenience (because it’s fast and was easy to simply invoke the model creation logic as a subroutine – Update LOS Model).
8. Possible additional solution quality metrics include initial vs. final GCP distribution metrics but are not implemented in the baseline version.
9. The heritage ALIAS (and Landsat) LOS model correction algorithms required the L1R file as input data so as to provide the L1R input space image dimensions. This information will now be available in the LOS model in the image sub-model so the L1R input is no longer required.

7.2.4 OLI Resampling Algorithm

7.2.4.1 Background/Introduction

The Operational Land Imager (OLI) resampling method is used to take a L1R image, which has unevenly spaced pixels with respect to the surface of the object imaged, and creates a reprojected image where all image pixels are located within an evenly spaced set of grid points, or output space, with respect the object imaged. This mapping is subject to the errors associated with the interpolation method used to determine the digital numbers associated with the output image.

The geometric resampling grid and the geometric model are used to calculate the mappings between the input and output space. The geometric model contains the individual detector offsets from a nominal location while the geometric resampling grid contains all other mapping variables. The resampling grid provides a mapping from a 2D input space to a 3D output space and vice versa. The output space corresponds to x/y/z projection locations while the input space corresponds to line/sample locations within the L1R. The z component in output space is elevation. If elevation is not to be accounted for during processing an elevation of zero is used for mapping output pixels to input pixels.

Due to what can be rather large sample-to-sample offsets within a L1R image, the cubic convolution interpolation option works in a two step process. A hybrid set of pixels in the sample direction are created using cubic convolution resampling in the line direction. This creates a set of unevenly spaced pixels in the sample direction. The Akima A interpolation method is then used to determine the final digital number for the output image by resampling the hybrid pixels in the sample direction. The nearest neighbor resampling option simply determines the closest input pixel for corresponding output pixel.

The OLI resampling algorithm is derived from the corresponding ALI algorithm used in ALIAS. The sensor architecture between the instruments is similar enough that a majority of the ALIAS algorithm can be reused. The baseline geometric modeling approach assumes that the 3D gridding approach used within ALIAS can also be used for OLI. The heritage algorithm will have to be modified to accommodate LDCM data formats.

7.2.4.2 Dependencies

The OLI resampling algorithm assumes that the Ancillary Data Preprocessing, Line-of-Sight (LOS) Model Creation and Line-of-Sight Projection to Ellipsoid and Terrain algorithms have been executed and a L1R has been generated. If a digital elevation model (DEM) is given as input to account for relief, or terrain, displacement the grid must have an adequate number and range (elevation bounds) of z-planes to cover the entire elevation range within the L1R. A geometric model and grid must be available for the L1R. More information about the data structure and contents of the Geometric Model and Resampling Grid can be found in the Ancillary Data Preprocessing, Line-of-Sight (LOS) Model Creation and Line-of-Sight Projection to Ellipsoid and Terrain Algorithm Description Documents (ADDs).

7.2.4.3 Inputs

The resampling algorithm and its component sub-algorithms use the inputs listed in the following table. Note that some of these "inputs" are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs
L1R Image
Resampling Grid (see the Line of Sight Projection ADD for contents)
Bands to process
Terrain correction Flag (yes/no)
DEM (if terrain flag set to yes)
SCA combine flag (yes/no)
Geometric model (see Line of Sight Model Creation ADD for contents)
Resampling type (CC,NN)
Minimum and maximum DN of output (see note #9)
Output data type
α (if resampling type is CC) (defaults to -0.5)
Fill pixel value

7.2.4.4 Outputs

Resampled output image (L1G, L1GT or L1T)
Resampled image data (band separated, either SCA combined or SCA separated)
Image data metadata fields (See tables 1 and 2)

7.2.4.5 Options

Cubic convolution or nearest neighbor resampling
Creating an output image with Sensor Chip Assemblies (SCAs) combined or separated
Applying terrain correction, yes or no

7.2.4.6 Procedure

OLI resampling interpolates radiometrically corrected but geometrically raw image data to a map projected output space. The resampling process uses information stored in the resampling grid along with focal plane calibration data stored in the geometric model to map output projection locations to an input location. Since an input location for an output pixel typically lies at a non-integer location interpolation is used to find the pixel values associated with this non-integer location. OLI resampling is performed on the geometrically raw L1R data using one of two methods; cubic convolution combined with the Akima A method, or nearest neighbor. Note that modulation transfer function compensation (MTFC) and bilinear resampling are not supported in the baseline algorithm. Due to the lack of inherent band registration and the need to perform sub-pixel registration to achieve OLI band alignment, cubic convolution combined with the Akima A interpolation method will be used to generate the standard LDCM products. It is also important to have the best subpixel accuracy in the output image during geometric characterization and calibration, so cubic convolution is chosen for interpolation during the characterization and calibration of the OLI instrument. The ALIAS-heritage nearest neighbor interpolation capability is also provided as an option for special-purpose science products and testing purposes. Since both standard product generation and geometric

characterization and calibration are the focus of this document, the only interpolation method discussed in detail here is the cubic convolution combined with the Akima A method.

During resampling, there is a need to know what input pixel goes with a given output pixel. The OLI geometric processing system does not have a "true" inverse model to perform this calculation. Instead, for a given output pixel, the corresponding input pixel is found from the forward and inverse mapping coefficients stored in the resampling grid. There are two scenarios when performing this calculation. The first involves performing resampling for a systematic image in which case the dimension for z, or elevation, is either zero or a constant value. This involves only a two dimensional operation in line and sample. The second involves performing resampling for a terrain corrected image. A terrain corrected image has the effects of relief removed from the output imagery. When working with a terrain corrected image, a 3-dimensional operation is performed during the inverse mapping with the dimensions being input (L1R) line, input sample, and elevation (figure 1). Both procedures of mapping output pixel locations to input pixel locations are discussed below.

Due to layout of the OLI focal plane, there are along-track offsets between spectral bands within each SCA, along-track offsets between even and odd SCAs, and a reversal of the band ordering in adjacent SCAs. This leads to an along-track offset in the imagery coverage area for a given band between odd and even SCAs as well as an offset between bands within each SCA. To create a more uniform image coverage within a geometrically corrected output product, the leading and trailing imagery associated with these offsets is trimmed. This trimming is controlled by a set of latitude/longitude bounds for the active image area for each band, contained in the input resampling grid. Trimming is implemented by converting these bounds to a look up table that lists the starting and ending sample location of active (non-fill) data for each line of the output image.

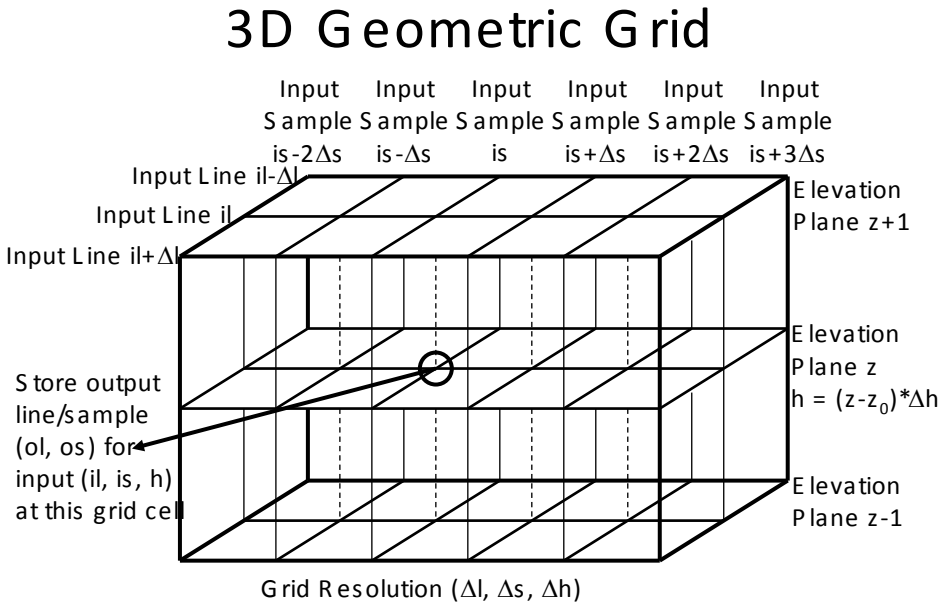


Figure 1. 3D Grid Representation

7.2.4.6.1 Using the geometric grid to map an output pixel location to an input pixel location.

To find an input line/sample location for an output line/sample location given that the elevation is zero:

1) Calculate an input line and sample location using the rough polynomial stored in the resampling grid and the current output line and sample location.

$$\text{approximate input line} = \sum_{n=0}^{N}\left[\sum_{m=0}^{M}\left(ra_{n,m} * (\text{outputsample})^{m}\right) * (\text{outputline})^{n}\right]$$

$$\text{approximate input sample} = \sum_{n=0}^{N}\left[\sum_{m=0}^{M}\left(rb_{n,m} * (\text{outputsample})^{m}\right) * (\text{outputline})^{n}\right]$$

Where:

ra = rough polynomial mapping coefficients for line mapping
rb = rough polynomial mapping coefficients for sample mapping
M = Number of sample coefficients in polynomial
N = Number of line coefficients in polynomial

Previous experience when working with the ALI instrument has demonstrated a 1st order polynomial in both the line and sample direction will suffice for the rough polynomial, thus M = N = 1.

$$\text{approximate input line} = a_0 + a_1 * \text{outputsample} + a_2 * \text{outputline} + a_3 * \text{outputsample} * \text{outputline}$$

$$\text{approximate input sample} = b_0 + b_1 * \text{outputsample} + a_2 * \text{outputline} + a_3 * \text{outputsample} * \text{outputline}$$

There is no evidence to believe that this will not also be the case when working with the OLI instrument.

2) Calculate the grid cell location for the approximate input line and sample location.

$$\text{row} = \frac{\text{approximate input line}}{\text{number of lines per cell}}$$

$$\text{column} = \frac{\text{approxinate input sample}}{\text{number of samples per cell}}$$

Where:
number of lines per cell = size of grid cell in lines
number of samples per cell = size of grid cell in samples

Set this grid cell column and row location as the current grid cell column and row location.

3) Using the current grid cell location check if the correct grid cell has been found.

Use input (current) mapping grid cell coefficients (a_i and b_i) to map output line and sample to input:

input line = $b_0 + b_1 *$ output sample $+ b_2 *$ output line $+ b_3 *$ output line $*$ output sample
input sample = $a_0 + a_1 *$ output sample $+ a_2 *$ output line $+ a_3 *$ output line $*$ output sample

Calculate the grid cell location for this input line and sample location:

$$\text{new row} = \frac{\text{input line}}{\text{number of lines per cell}}$$

$$\text{new column} = \frac{\text{input sample}}{\text{number of samples per cell}}$$

If the new grid cell (new row and new column) is the same as the current grid cell (current row and current column):

The correct grid cell has been found, inverse grid mapping coefficients for this grid cell are used to calculate the input line/sample for the current output line/sample.

If the new grid cell (new row and new column) is not the same the current grid cell (current row and current column):

The new grid cell is chosen as current grid cell and the 3rd step is repeated until the correct grid cell is found.

This routine or function listed above, of mapping output pixel locations to input pixel locations without taking into account elevation, will be referred to as ols2ils (output space line-sample to input space line-sample mapping). The ols2ils sub-algorithm takes a given output line and sample location and calculates the grid cell column and row location along with the corresponding input line and sample location for that output location.

To find an input line/sample location for an output line/sample location given that the elevation is not zero:

1) Find the z planes that the elevation associated with the output pixel falls between.

$$z\ \text{plane} = (\text{int})\text{floor}\left(\frac{\text{elevation}}{\text{elevation increment}}\right) + z_{elev=0}$$

Where:

elevation = elevation associated with current output location (from DEM)
elevation increment = elevation increment between z planes stored in grid
$z_{elev=0}$ = zero z plane, the index of the zero elevation z-plane

The output line/sample falls between z plane and z plane+1.

2) Call ols2ils for z plane and z plane+1. This yields (input $sample_0$, input $line_0$), and (input $sample_1$, input $line_1$).

3) Interpolate between z plane and z plane + 1 to find input line and sample location for elevation.

Calculate elevations for z plane and z plane + 1:
$elev_0$ = elevation increment * (z plane - zero z plane)
$elev_1$ = $elev_0$ + elevation increment

Calculate weights for ols2ils results:

$$w_0 = \frac{elev_1 - elevation}{elev_1 - elev_0}$$

$$w_1 = \frac{elevation - elev_0}{elev_1 - elev_0}$$

input sample = input $sample_0$ * w_0 + input $sample_1$ * w_1
input line = input $line_0$ * w_0 + input $line_1$ * w_1
Where:
input $sample_0$ = input sample for z plane
input $sample_1$ = input sample for z plane + 1
input $line_0$ = input line for z plane
input $line_1$ = input line for z plane + 1

This routine or function listed above, which performs the three-dimensional output space line-sample to input space line-sample mapping, is referred to as 3d_ols2ils.

7.2.4.6.2 Resampling Methodology

The along and cross track detector offsets are applied during resampling. These include both the dynamic odd and even terrain-dependent relief and parallax effects that were calculated during the resampling grid generation, and the individual detector selection shift that are stored in the OLI geometric model. The nature of these geometric effects due to the individual detector characteristics is such that, in input space, they are evenly spaced in the line direction but unevenly spaced in the sample direction. This is due to the fact that as you move along raw imagery in the line direction, the detector number does not change. Since the detector number does not change along the line direction in raw input space, the along track detector offset, stored within the geometric model, does not change. These geometric effects, due to these detector offsets, are slowly varying in time staying essentially constant within the area that resampling is performed. Therefore the along track geometric effect, and essentially spacing in the line direction, can be treated as a constant over this area. The same logic helps explain why the across track detector offset is not constant in the sample direction, since each sample comes from a different detector. This creates unevenly spaced samples in raw input space. An example of a detector layout and its' associated offset can be seen in figure 2. The squares in figure 2 represent a location of an input pixel, taking into account the detector offsets. The circle with the cross hairs in figure 2 represents the true input location for the current output pixel. It is at this point that an interpolated value is needed to represent the current output pixel.

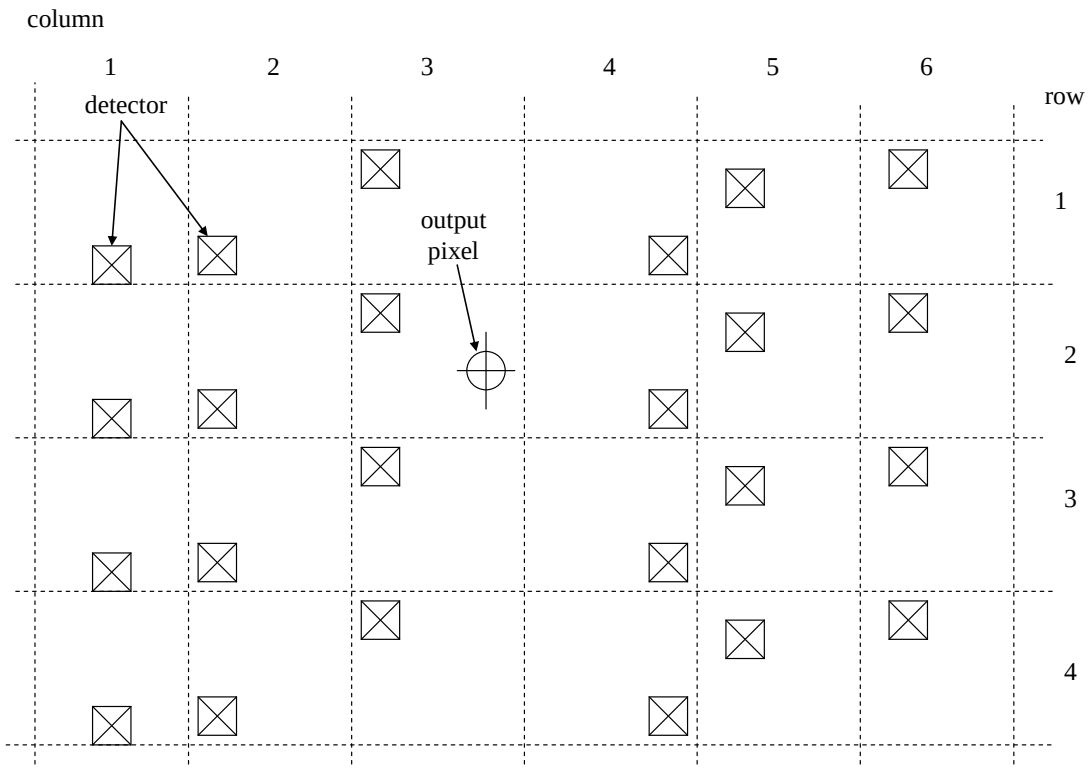


Figure 2. Example detector layout

Detector offsets are handled in the resampler by first applying a resampling kernel in the line direction that assumes evenly spaced detectors. Cubic convolution interpolation is used in the line direction; this will align a set of pixels in the sample direction. Once the pixels are aligned in the sample direction, at uneven spacing, the Akima A interpolation is used to find the final output pixel value.

Cubic convolution interpolation uses a set of piecewise cubic spline interpolating polynomials. The polynomials have this form:

$$f(x) = \begin{cases} (\alpha+2)|x|^3 - (\alpha+3)|x|^2 + 1 & 0 \le |x| < 1 \\ \alpha|x|^3 - 5\alpha|x|^2 + 8\alpha|x| - 4\alpha & 1 \le |x| < 2 \\ 0 & |x| > 2 \end{cases}$$

Four points, centered on the point to be interpolated, are used in interpolation. The weights for each point are generated from *f(x)*. The α in the cubic convolution function is a variable parameter that effects the edge slope of the function. For standard processing, a value of -0.5 is used. An example of what the cubic convolution function looks like, and the corresponding weights for a phase shift of zero (marked as x's), is shown in figure 3.

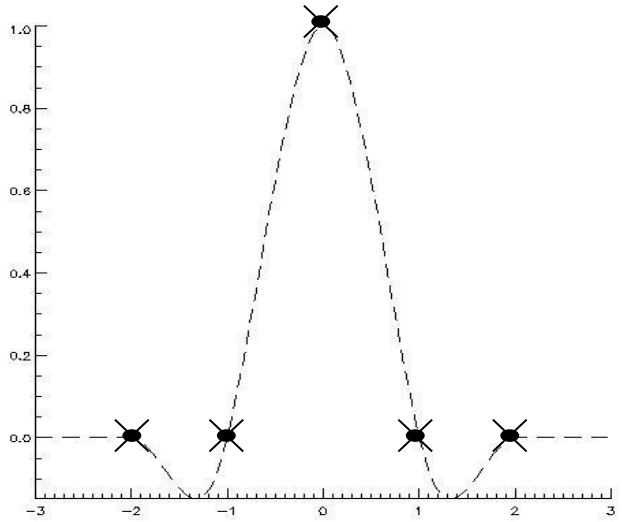


Figure 3. Cubic Convolution Function

As stated previously; for the OLI resampler the cubic convolution resampling process produces a set of hybrid points that are aligned in the line direction. This is done by resampling several sets of L1R pixels in the line direction using the cubic convolution kernel; each time cubic convolution is performed one hybrid pixel is produced. The set of hybrid points produced from the cubic convolution process are not evenly spaced in the sample direction. Figure 4 illustrates a set of hybrid samples that have been aligned in the line direction using the cubic convolution process.

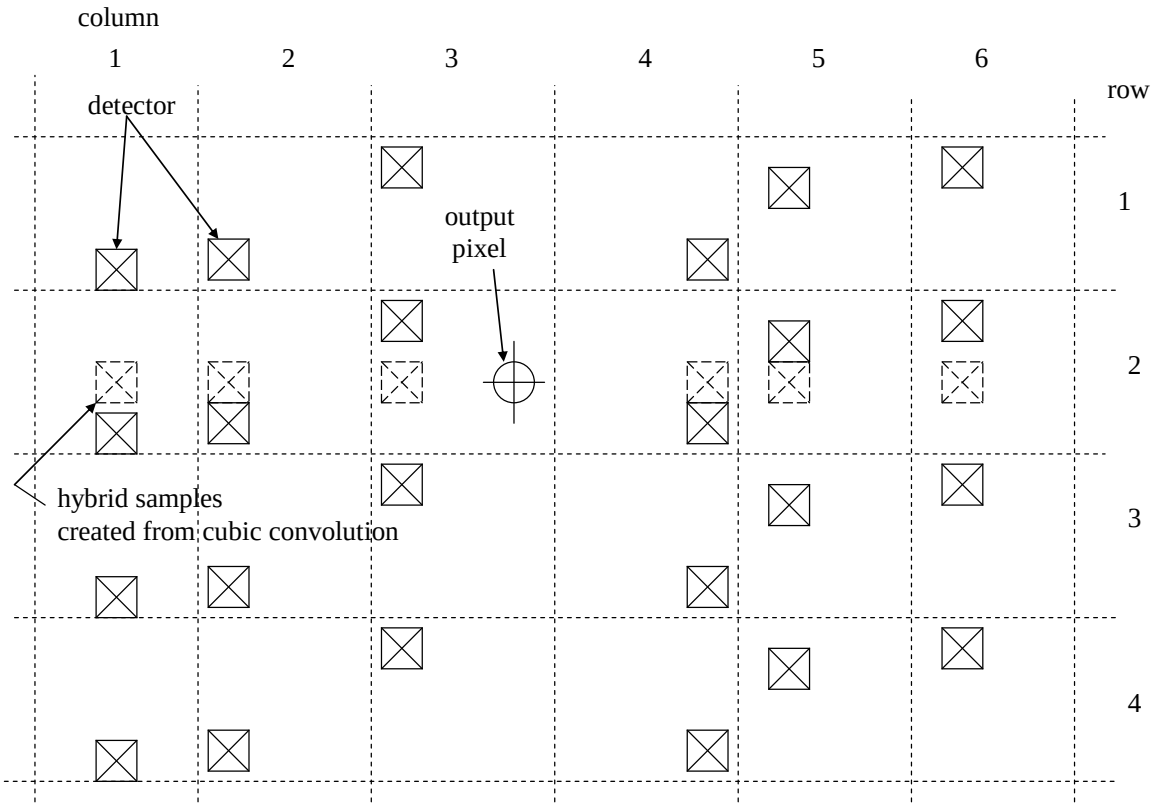


Figure 4. Hybrid pixels for detector offsets

The Akima A method for interpolation is used for interpolating the hybrid pixels created from the cubic convolution process. This method of interpolation does not require the samples used to be evenly spaced. The Akima A method uses a third order polynomial for interpolation. The interpolating polynomial is defined by the coordinates and the slopes of the two points that are on either side of the point to be interpolated. The slopes of the adjacent points are determined as follows:

If five points are defined as 1, 2, 3, 4, and 5 then the slope at point 3, t, is defined as:

$$t = \frac{|m_4 - m_3|m_2 + |m_2 - m_1|m_3}{|m_4 - m_3| + |m_2 - m_3|}$$

Where:

m_1 = slope of line segment defined by points 1 and 2
m_2 = slope of line segment defined by points 2 and 3
m_3 = slope of line segment defined by points 3 and 4
m_4 = slope of line segment defined by points 4 and 5

The Akima A method of interpolation is based upon the values (*y*) and slopes (*t*) on either side of the point that is to be interpolated. The interpolating polynomial for a point x between x_i and x_{i+1} is then defined as:

$$y = y_i + t_i * (x - x_i) + \frac{3\frac{y_{i+1} - y_i}{x_{i+1} - x_i} - 2t_i - t_{i+1}}{x_{i+1} - x_i} * (x - x_i)^2 + \frac{t_i + t_{i+1} - 2\frac{y_{i+1} - y_i}{x_{i+1} - x_i}}{(x_{i+1} - x_i)^2} * (x - x_i)^3$$

Where:

x = sample location of point to be interpolated
x_i = location of point to the left of x
x_{i+1} = location of point to the right of x
y_i = DN value for the input point at x_i
y = interpolated DN value for an output line and sample location

This methodology must be adjusted somewhat to account for higher frequency image distortion effects than those that can be captured by the conventional resampling grid. To model such effects, the LDCM attitude data stream is separated in to low-frequency and high-frequency segments with the low-frequency portion being used for the OLI line-of-sight projection operations that build the resampling grid. The high-frequency data are interpolated to match the OLI panchromatic band line sampling times and stored in the OLI LOS model in a jitter table for application as an extra correction at image resampling time. The process of separating the attitude data stream by frequency is described in the OLI Line-of-Sight Model Creation Algorithm Description Document.

Sensitivity coefficients that relate these high-frequency roll-pitch-yaw jitter terms to the equivalent input image space line and sample offset effects are stored in the OLI LOS grid. This makes it possible to look up the roll-pitch-yaw jitter for each image line being resampled, and convert the jitter values to compensating input line/sample corrections that are used to refine the image interpolation location coordinates. The generation of these sensitivity coefficients is described in the OLI Line-of-Sight Projection/Grid Generation Algorithm Description Document. The process by which the jitter table from the OLI model and jitter sensitivity coefficients from the OLI grid are used during image resampling is shown schematically in Figure 5 below. The items in green in the figure are new structures added to support jitter correction.

Since the jitter effects vary by image line, the time delay between even and odd (or deselected) detectors will lead to slightly different jitter effects in adjacent image samples. This is depicted below in Figure 6. Six time samples (t0 through t5) for six adjacent detectors are shown in the figure. Note that the input line location returned by the grid is adjusted differently for the even and odd detectors due to their timing offset. Including the effects of detector deselect, the interpolated line location for the hybrid pixels could be different for each detector. The current approach does not account for sample-to-sample variations in jitter for each detector, applying the jitter correction only at the output location. This preserves the uniform along-track sampling assumption required to apply the cubic convolution kernel. Also note that while it is the interpolation location that is adjusted relative to the input pixel locations in the line direction, it is the detector sample locations that are adjusted relative to the interpolation location in the sample direction. The jitter-adjusted resampling procedure is explained in more detail below.

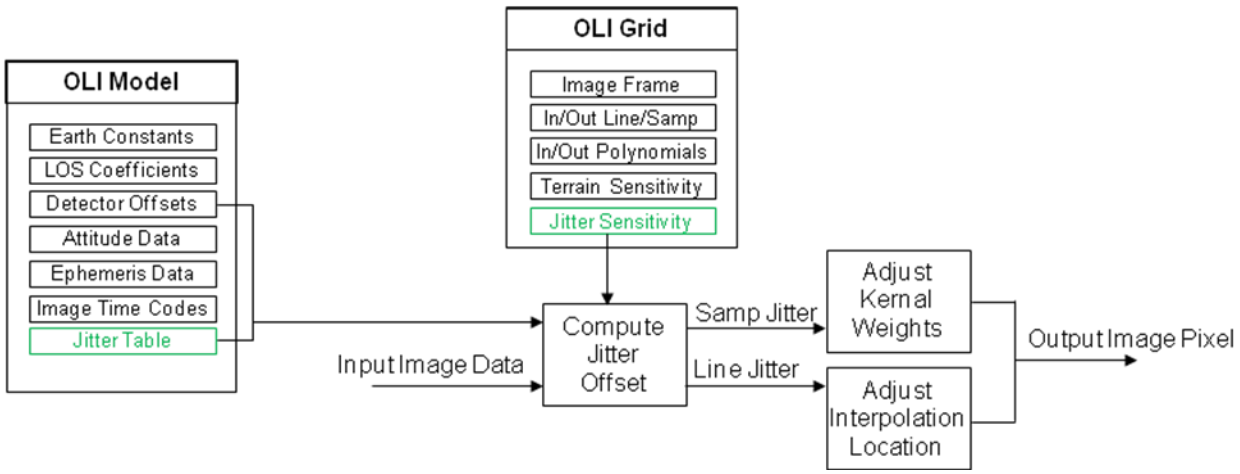


Figure 5: OLI LOS Model and OLI LOS Grid Jitter Correction Data Flow

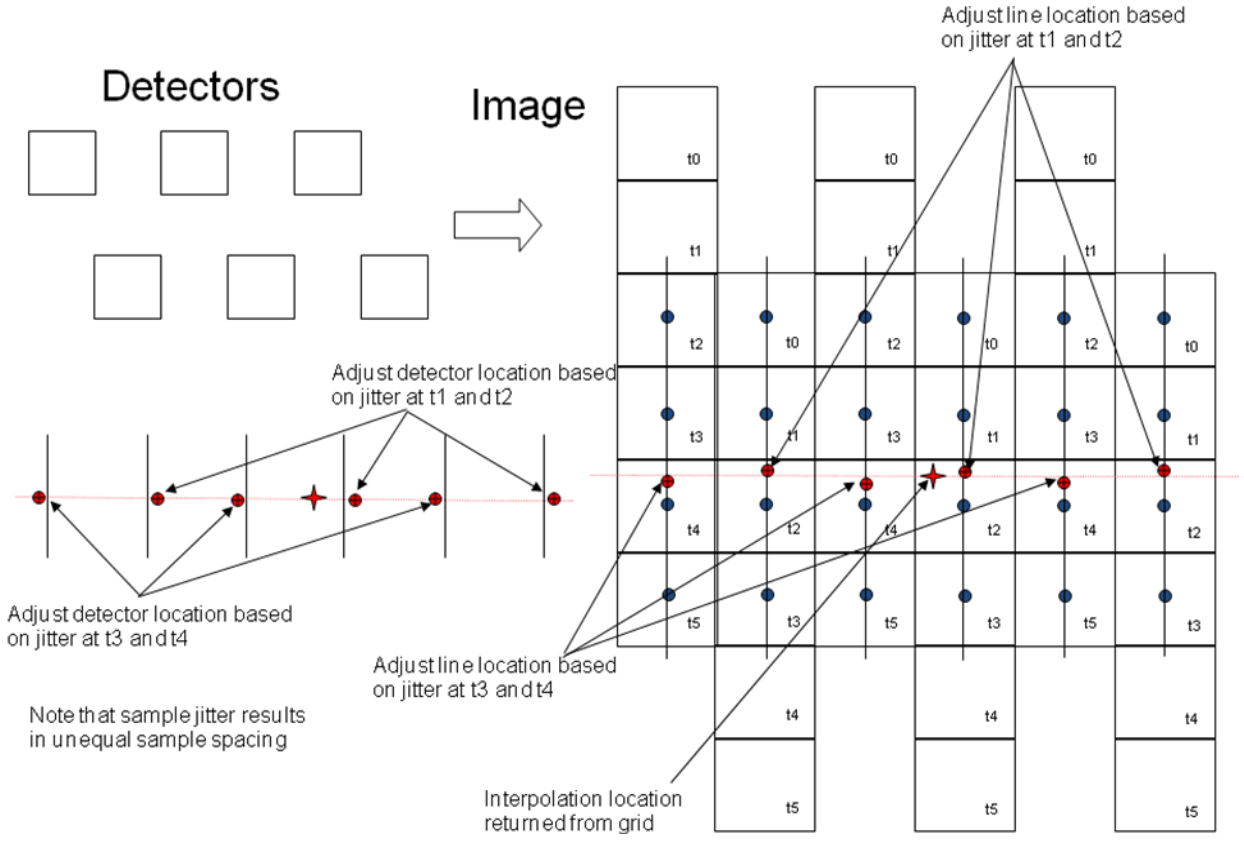


Figure 6: Jitter Effects in Image Resampling

7.2.4.6.3 Building The SCA-trimmed Look Up Table (LUT).

Allocate SCA-trim LUT. There is a starting and ending sample location of active or valid imagery stored for each line of output in the SCA-trimming look up table.

LUT = malloc(2 * nl)

Where nl = number of lines in output imagery

Given the set of geographic corner coordinates, read from the input grid file, that represent valid imagery for a given band:

1. Map four corners to output projection coordinates.
2. Map four output projection coordinates to line and sample coordinates.
3. Set up polygon definition from four coordinates:
   <px0,py0> = <sample upper left, line upper left>
   <px1,py1> = <sample upper right, line upper right>
   <px2,py2> = <sample lower right, line lower right>
   <px3,py3> = <sample lower left, line lower left>
   <px4,py4> = <sample upper left, line upper left>
4. Set up sample locations for each line that is outside active imagery:
   osamp1 = -1.0
   osamp2 = output number of samples
   for nn = 0 to 3
   if px[nn] < osamp1 then osamp1 = px[nn] – 1.0
   if px[nn] > osamp2 then osamp2 = px[nn] + 1.0
5. Initialize LUT values to fill for all output lines:
   For nn = 0 to (2 * number of output lines)
   LUT[nn] = 0
6. For nn = 0 to number of output lines (nn and current line are synonymous).
   6.1. Define line by sample locations calculated from 4 and current line
   <x0,y0> = <osamp1, nn>
   <x1,y1> = <osamp2, nn>
   6.2. Determine intersection between sides of polygon defined in 3 and line defined in 6.1
   Initialize number of intersections for current line:
   intersections = 0
   For nn = 0 to 3
   (Simple line intersection routine)
   xlk = x0 – x1
   ylk = y0 – y1
   xnm = px[nn] – px[nn+1]
   ynm = py[nn] – py[nn+1]
   xmk = px[nn+1] – x1
   ymk = py[nn -1] – y1
   det = xnm * ylk – ynm * xlk
   if (| det | <= TOL) lines are parallel, no intersection found.
   s = (xnm * ymk - ynm * xmk) / det
   t = (xlk * ymk - ylk * xmk) / det
   if(s<0.0 || s>1.0 || t<0.0 || t>1.0)
   no intersection found
   else
   intersection found, calculate point:
   xp[intersections] = x1 + xlk * s
   yp[intersections] = y1 + ylk * s
   intersections++
   6.3. If number of intersections from 6.2. is two then the current line has valid active imagery and the look up table values are these intersections and represent the start and stop of valid imagery. Store values in SCA-trim lookup table.

```
if   xp[0] > xp[1]
        LUT[ 2 * nn ]     = xp[1]
        LUT[ 2 * nn + 1] = xp[0]
else
        LUT[ 2 * nn ]     = xp[0]
        LUT[ 2 * nn + 1] = xp[1]
```

(Note: If number of intersections is not two then current line has no valid active imagery and SCA-trim lookup table will contain points outside of imagery, fill will be used).

7.2.4.6.4 Load/Build Information

To resample a Level 1R data set, the image file, grid file, geometric model, and, if the effects terrain are to be removed, a DEM must be opened. See note #3

7.2.4.6.5 Resample Level1R Imagery

Loop on each band of each SCA for resampling.

1. Get resampling grid for the band and SCA to be processed.
2. Build SCA-trimming table.
3. Read one band of imagery for one SCA. Note #7.
   - 3.1. Initialize jitter correction parameters
     If current band is panchromatic then jitter_scale = 1
     Otherwise jitter_scale = 2
4. Loop on output line/samples
   - 4.1. Check to see if output line/sample is within SCA-trimming bounds.
     if output sample > LUT[2 * output line] &&
     output sample < LUT[2 * output line + 1] then proceed
     else output pixel = fill
   - 4.2. If image is terrain corrected, calculate elevation dependent input line/sample location.
     4.2.1) Get elevation for output pixel location X/Y location from DEM (elevation). See note #3.
     4.2.2) Map the output line/sample back into input space using the grid and the function 3d_ols2ils.
   - 4.3. If image is not terrain corrected calculate zero elevation (ellipsoid surface) input line/sample location.
     4.3.1) Set elevation to zero
     4.3.2) Map the output line/sample back into input space using the grid and the function ols2ils.
   - 4.4. Calculate actual input sample location; for sample location (int)input sample calculated from either 4.2 or 4.3:
     4.4.1) Calculate detector offset parallax scale.
     Scale = (int) floor(detector along track offset + 0.5) (in geometric model). See note #4.
     4.4.2) Calculate sample odd/even parallax offset
     Δsample_oe = (d_0 + elevation * d_1) * scale
     Note that (d_0 + elevation * d_1) is the parallax (in pixels) per pixel of along track offset from the nominal detector location.
     Where:
     $d_{0,1}$ = odd/even sample parallax coefficients stored in the grid
     4.4.3) Get sample fractional offset
     fractional sample offset =

detector across track offset (in geometric model)

4.4.4) Calculate sample jitter adjustment

4.4.4.1) Calculate the index into the jitter table for the current image line

jit_index = (int)(jitter_scale*(input line – pixel column fill (defined below)))

Make sure jitter index is within the range of the jitter table. Set to the min or max value (whichever is closest) if it is outside the range.

4.4.4.2) Calculate the fractional jitter table index

Δjit_index = jitter_scale * input line – floor(jitter_scale * input line)

4.4.4.3) Calculate simple sample jitter adjustment

```
samp_jitter0 = samp_sens[0] * jitter_table[jit_index].roll
             + samp_sens[1] * jitter_table[jit_index].pitch
             + samp_sens[2] * jitter_table[jit_index].yaw
samp_jitter1 = samp_sens[0] * jitter_table[jit_index+1].roll
             + samp_sens[1] * jitter_table[jit_index+1].pitch
             + samp_sens[2] * jitter_table[jit_index+1].yaw
```

samp_jitter = samp_jitter0 * (1-Δjit_index) + samp_jitter1*Δjit_index

Where:

samp_sens[0] is the sample direction jitter roll sensitivity,
samp_sens[1] is the sample direction jitter pitch sensitivity,
samp_sens[2] is the sample direction jitter yaw sensitivity,
for the current grid cell, from the OLI grid.
jitter_table[n] is the jitter table roll-pitch-yaw vector for row n,
from the OLI model.

4.4.4.4) Refine the sample jitter to compensate for line jitter

```
line_jitter0 = line_sens[0] * jitter_table[jit_index].roll
             + line_sens[1] * jitter_table[jit_index].pitch
             + line_sens[2] * jitter_table[jit_index].yaw
line_jitter1 = line_sens[0] * jitter_table[jit_index+1].roll
             + line_sens[1] * jitter_table[jit_index+1].pitch
             + line_sens[2] * jitter_table[jit_index+1].yaw
```

line_jitter = line_jitter0 * (1-Δjit_index) + line_jitter1*Δjit_index

Where:

line_sens[0] is the line direction jitter roll sensitivity,
line_sens[1] is the line direction jitter pitch sensitivity,
line_sens[2] is the line direction jitter yaw sensitivity,
for the current grid cell, from the OLI grid.

This is the error in the line coordinate used above, due to line jitter.

```
samp_rate =
   samp_sens[0]*(jitter_table[jit_index+1].roll-jitter_table[jit_index].roll)
+  samp_sens[1]*(jitter_table[jit_index+1].pitch-jitter_table[jit_index].pitch)
+  samp_sens[2]*(jitter_table[jit_index+1].yaw-jitter_table[jit_index].yaw)
```

This is the rate of change of sample jitter with line coordinate.

samp_jitter += line_jitter*samp_rate

This is the sample jitter correction adjusted for the effects of line jitter.

4.4.5) actual input sample = input sample - Δsample_oe - samp_jitter - fractional sample offset (See note #5). These corrections are subtracted rather than added because what we are doing here is, rather than adjusting the input space interpolation location, computing the apparent location of the detector to the left of the interpolation location to make sure we have the correct range of samples to feed the interpolation logic. If the above adjustments lead to the "actual input sample" being greater than (to the right of) the original input sample location, then we move our sample range one more sample to the left. We perform a similar calculation on the detector to the right of the input space interpolation location to make sure that we don't have to shift one more sample in that direction. See also the note in section 4.6.2 below.

4.5. Create fractional pixel shift for current input location:

Δline = input line - (int) input line

Δsample = input sample - (int) input sample

4.6. Create aligned samples for Akima resampling by applying cubic convolution weights in line direction.

4.6.1. Loop on actual input sample location:

For hybrid sample = (int) actual input sample - 2 to (int) actual input sample +3 (Note #5. One extra hybrid sample created to left and right of minimum number of samples needed for Akima interpolation)

In the case of NN resampling, the loop limits are reduced to:

For hybrid sample = (int) actual input sample to (int) actual input sample +1

4.6.1.1. Calculate line and hybrid sample detector offset parallax scale

scale = (int) floor(detector along track offset + 0.5) (in geometric model). See note #4.

4.6.1.2. Calculate odd/even detector offset, parallax correction, and jitter correction for hybrid detector.

4.6.1.2.1. Odd/even detector offset and parallax corrections.

Δline_oe = (c_0 + elevation * c_1) * scale + pixel column fill - nominal detector fill - at_offset[hybrid sample]

Δsample_oe = (d_0 + elevation * d_1) * scale

Where:

$c_{0,1}$ = odd/even line parallax coefficients stored in the grid

$d_{0,1}$ = odd/even sample parallax coefficients stored in the grid. Note that (c_0 + elevation * c_1) is the along-track parallax (in pixels) per pixel of along-track offset from the nominal detector location and (d_0 + elevation * d_1) is the across-track parallax (in pixels) per pixel of along-track offset from the nominal detector location.

4.6.1.2.2. Jitter correction

The sample jitter correction is calculated as described in section 4.4.4 above. The line jitter correction is calculated as follows:

jit_index = (int)(jitter_scale*(input line – pixel column fill))

Δjit_index = jitter_scale * input line – floor(jitter_scale * input line)

line_jitter0 = line_sens[0] * jitter_table[jit_index].roll
+ line_sens[1] * jitter_table[jit_index].pitch
+ line_sens[2] * jitter_table[jit_index].yaw

line_jitter1 = line_sens[0] * jitter_table[jit_index+1].roll
+ line_sens[1] * jitter_table[jit_index+1].pitch
+ line_sens[2] * jitter_table[jit_index+1].yaw

line_jitter = line_jitter0 * (1-Δjit_index) + line_jitter1*Δjit_index

Where:

line_sens[0] is the line direction jitter roll sensitivity,
line_sens[1] is the line direction jitter pitch sensitivity,
line_sens[2] is the line direction jitter yaw sensitivity,
for the current grid cell, from the OLI grid.
This is the error in the line coordinate due to jitter.
line_rate =
line_sens[0]*(jitter_table[jit_index+1].roll-jitter_table[jit_index].roll)
+ line_sens[1]*(jitter_table[jit_index+1].pitch-jitter_table[jit_index].pitch)
+ line_sens[2]*(jitter_table[jit_index+1].yaw-jitter_table[jit_index].yaw)
This is the rate of change of line jitter with line coordinate.
line_jitter += line_jitter*line_rate
This is the line jitter correction adjusted for the second order effects of line jitter. Note the similarity to the sample correction described in 4.4.4.4.

4.6.1.3. Calculate new hybrid line location.
4.6.1.3.1. hybrid line = (int)floor(input line + Δline_oe + line_jitter) .
Note that in this case we add the corrections since we are adjusting the interpolation location.
4.6.1.4. Calculate new fractional hybrid line location.
Δhybrid line = input line + Δline_oe +line_jitter – hybrid line
If |Δhybrid line| > 1 then the integer line index must be adjusted and Δhybrid line brought back into the -1 < Δhybrid line < 1 range (see note #5).
4.6.1.5. Apply cubic convolution in line direction to hybrid sample line DNs.
4.6.1.5.1. Calculate cubic convolution weights. See note #2

$$w_{n+2} = \sum_{n=-1}^{2} f(n - \Delta\text{hybrid line})$$

Where f is equal to cubic convolution function.
4.6.1.5.2. Apply cubic convolution weights to L1R DNs.
hybrid line DN = $w_0 * h_0 + w_1 * h_1 + w_2 * h_2 + w_3 * h_3$
Where
w_0, w_1, w_2, w_3 = Cubic convolution weights for Δhybrid line.
h_0 = DN from L1R for hybrid sample, input line location - 1
h_1 = DN from L1R for hybrid sample, input line location
h_2 = DN from L1R for hybrid sample, input line location + 1
h_3 = DN from L1R for hybrid sample, input line location + 2
In the case of NN resampling, the values of hybrid line and Δhybrid line are used to select the closest line for the current detector/sample column, instead of being used to compute weights. The hybrid line DN is then the L1R DN value for the closest line location.
4.6.2. Calculate the apparent Akima pixel location for the current hybrid sample.
Akima pixel location x_i =
hybrid sample location - Δsample_oe
- across track detector offset (in geometric model)
- samp_jitter (computed per section 4.6.1.2.2 above)
Note that in this case the across-track terrain parallax and sample jitter effects are subtracted instead of added. This is because we are adjusting the apparent detector location relative to the output pixel interpolation point instead of adjusting the output pixel interpolation location itself. We must do it this way in the sample direction because the adjustments are different for each detector. As for the across-track offset term, which is also unique for each detector, the

detector offset corrections are designed to be applied as line-of-sight corrections in the instrument coordinate system. As such, the along-track offset is a +X LOS correction and the across-track offset is a +Y LOS correction. The instrument +X axis is in the +line direction but the +Y axis is in the –sample direction, so this correction is also subtracted from the apparent detector location.

4.7. Calculate output DN using Akima interpolation and hybrid line/sample information from 4.6.1 and 4.6.2.

4.7.1. Calculate Akima weights according to pixel locations from 4.6.2.

$$m_0 = \frac{\mathrm{DN}_1 - \mathrm{DN}_0}{x_1 - x_0}$$

$$m_1 = \frac{\mathrm{DN}_2 - \mathrm{DN}_1}{x_2 - x_1}$$

$$m_2 = \frac{\mathrm{DN}_3 - \mathrm{DN}_2}{x_3 - x_2}$$

$$m_3 = \frac{\mathrm{DN}_4 - \mathrm{DN}_3}{x_4 - x_3}$$

$$m_4 = \frac{\mathrm{DN}_5 - \mathrm{DN}_4}{x_5 - x_4}$$

$$ak_0 = \mathrm{DN}_2$$

$$ak_1 = \frac{|m_3 - m_2| * m_1 + |m_1 - m_0| * m_2}{|m_3 - m_2| + |m_1 - m_0|}$$

$$ak_2 = \frac{(3.0 * m_2 - 2.0 * ak_1 - \frac{|m_4 - m_3| * m_2 + |m_2 - m_1| * m_3}{|m_4 - m_3| + |m_2 - m_1|}}{x_3 - x_2}$$

$$ak_3 = \frac{ak_1 + \frac{|m_4 - m_3| * m_2 + |m_2 - m_1| * m_3}{|m_4 - m_3| + |m_2 - m_1|} - 2.0 * m_2}{(x_3 - x_2)^2}$$

Where:

DN_n = hybrid DNs calculated from cubic convolution, step 4.6.1.

x_n = Akima locations calculated in step 4.6.2.

ak_n = Akima weights

m_n = Akima slopes

4.7.2. Calculate output pixel DN using Akima A method.

$$\mathrm{output\,DN} = ak_0 + ak_1 * ds + ak_3 * ds^2 + ak_4 * ds^3$$

Where

$$ds = (\Delta\mathrm{sample} + x_2)$$

The output sample point is located between hybrid samples x_2 and x_3 where x_n is from n=0…5. In the case of NN resampling, the Akima pixel locations for the two closest detectors are examined to see which is closest to the output location. The hybrid line DN value for the closest detector is selected as the output DN value.

4.8. Write output DN to image file. See note #9.

5. Write out data descriptor record for image file. The baseline contents of the data descriptor record are shown in table 1. All fields present in the table refer to the imagery associated with the DDR unless otherwise specified. Note that the scene roll angle is a new field added for off-nadir acquisitions. It would be computed from the LOS model by interpolating the roll angle from the "original" attitude data sequence at the time corresponding to the precision model reference time t_ref. This would be done using the logic described in the Find Attitude sub-algorithm in the LOS Projection ADD, except operating on the "original" rather than the "corrected" attitude data sequence. The logic for using the "original" data is so that this scene roll value will not change due to LOS model correction. The sign convention on the roll angle is such that a positive roll angle would correspond to a positive orbital Y coordinate which is looking to starboard (See note 11).

7.2.4.6.6 Combining SCAs into one output file.

For an SCA combined output image the overlap region between SCAs can be handled by averaging the pixels between SCAs (See Note 12).

7.2.4.7 Prototype Code

Input to the executable is an ODL file, output is a HDF5 file containing the image data and corresponding metadata. The output format follows the format of the L1G DFCB version 1.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall -march=nocona -m32

Main driver for resampler (oliresample)

Main driver for OLI resampler. Performs the following steps or calls the following modules.

1) Read input ODL parameters (getpar).
2) Read OLI input file (oli_get_model).
3) Read OLI grid headers (oli_get_grid_headers).
If terrain correction read DEM file (oli_get_dem).
4) Open L1G image file (open_l1g_resamp_image).
5) Get fill pixel value (get_fill_pixel).
For each band to process
6) Read grid band pointers (oli_get_grid_pointers).
7) Open/initialize L1G band file (start_l1g_resamp_band).
8) Setup resampling kernel (Kernal_Setup).
9) Read resampling kernal information for resampling (get_kernal_info).
For each SCA
10) Read one SCAs worth of data from L0ra (get_input_image_data_l0ra).
11) Resample SCA worth of data (resample_image).
if not SCA combined image file write SCAs worth of data (write_l1g_resamp_band).
If SCA combined image file write full SCA file (write_l1g_resamp_band).
12) Close band in L1G output file (stop_l1g_resamp_writing_band).
13) Free grid band pointer (oli_free_grid).
14) Close L1G image file (close_l1g_resamp_image).

15) Update L1G metadata (update_l1g_metadata).

Get resampling processing parameters (getpar).
This function reads the OLI resampling parameters from the ODL file. Also contains two functions, get_combine_sca and get_fill_pixel, that will return input flags as to whether 1) combine the SCAs in the output image and 2) what DN value should be used for fill.

Resample a given set of DN value using the Akima method (akima).
Function takes a given set of X locations with corresponding Y values and finds the Y value for the given input X location (xp). Function returns interpolated Y value associated with coordinate xp.

Calculate cubic convolution weight for a given location (cubic_convolution).
Given a cubic convolution alpha parameter and X value return the Y value associated with the cubic convolution function.

For a given band read one SCAs worth of L0R imagery (get_input_image_data_l0ra).
Given a L0rp file name, band number, and SCA number read an SCAs worth of data from L0rp file. Number of lines to read is taken from number of lines stored in models image data structure.

Set up resampling kernel (module kernal.c).
Using a set of functions, create a set of resampling weights. The resampling kernel is created and managed though several steps within the kernal.c file.

Kernal_Setup sets up kernal table or pointer. Allocates pointer and calls Create_Resampling_Kernal_1D to create a set of cubic convolution weights. Set is a look-up table of 1D cubic weights representing 1/64 of a shift in pixel locations.

Cleanup_Kernal frees up cubic convolution pointer.

Create_Resampling_Kernal_1D creates a set of one dimensional cubic convolution based on the input alpha parameter.

Get_Resample_Weight_Table_Ptr returns a pointer containing a set of 1D cubic convolution weights.

get_lines_in_kernal returns number of lines in resampling kernal.

get_samples_in_kernal returns number of samples in resampling kernal.

num_left_kernal_samples returns number of resampling weights to the "left" of the point that is to be interpolated.

num_right_kernal_sample returns number of resampling weights to the "right" of the point that is to be interpolated.

num_top_kernal_lines returns the number of lines "above" the point to be interpolated.

num_bottom_kernal_lines returns the number of lines "below" the point to be interpolated.

get_kernal_step_size returns the offset size in pixels between two sets of resampling weights.

get_kernal_info returns the number of steps (or number of sets of weights) within the resampling kernal, total number of sets of weights within the resampling table, width of resampling kernal, and height of resampling kernal.

Read DEM file (oli_get_dem).
Reads (Image Processing Element) IPE L1G file contain DEM data.

Open, close, write to L1G output image file (file output_image_data.c)

The file output_image_data.c contains several routines used for managing the output L1G file. Calls and functions are listed below.

open_l1g_resamp_image opens a L1G file.
start_l1g_resamp_band opens one band within an L1G file.
write_l1g_resamp_band writes image data to L1G file.
stop_l1g_resamp_writing_band closes band within L1G file.
close_l1g_resamp_image closed L1G file.

Resample one SCA for one band of L0Rp imagery (file resample_image.c)

The file resample.c contains several functions used in resampling imagery.

setup_trim_lut builds a lookup table that contains the starting and ending output pixel of valid imagery. Everything outside of this bounds will be set as fill

cleanup_trim_lut frees static buffer that contains SCA-trimming lookup table array.

get_kernal_info retrieves resampling weight table and corresponding characteristics.

setup_detector_offsets stores the detector offsets, along and across, level-0R fill, and nominal detector fill within arrays. Used by resample_image for applying detector offsets when resampling imagery.

resample_image is the main guts of the resampler. Takes the image data, DEM data if terrain corrected, grid band pointer, and OLI model structure to resample one SCA or one band of imagery. Loops on output pixels mapping each output pixel location to a input location and resamples L0Rp (or L1R when it becomes available) using algorithm described in procedure section.

calc_jitter computes the sample and line direction jitter corrections for the current input line/sample location. This corrections are the adjustments to the input space interpolation location required to compensate for the high frequency jitter present at the time of observation.

calc_jitter_samp is a simplified version of calc_jitter that computes only the sample direction jitter correction. It is implemented as a separate function for processing efficiency because it is invoked more frequently than calc_jitter.

Update L1G metadata information (update_l1g_metadata).

Update L1G metadata according to projection information stored within resampling grid.

Write out ENVI header file (write_envi_hdr).

Writes out ENVI header file for image flat file that is written to disk. Only used for testing purposes.

Input and Output File Details

Output is a L1G image file formatted according to the L1G DFCB. The output is a HDF5 file . The metadata associated with the output file is listed below. These tables follow the meta data fields in version 1 of the LDCM Level-1 G DFCB. The metadata is split up into a file metadata and band metadata. For further information on this format see the L1G DFCB. Not all fields within the prototype metadata fields are filled in with valid data. Fields in which data is *not* correctly filled are indicated in italics (see notes #9 and #10).

File Metadata

Field	Description	Type
Projection Code	GCTP projection code	integer
Zone Code	Map projection zone code	integer
Datum	Projection datum code	char[16]

Spheroid Code	Projection spheroid code	integer
Projection Units	Map projection units	char[12]
Projection Parameters	GCTP projection parameters	double[15]
WRS Path	*WRS-2 Path*	*integer*
WRS Row	*WRS-2 Row*	*integer*
Roll Angle	*Off nadir pointing angle*	*double*
Spacecraft	*Spacecraft name*	*char[32]*
Collection Type	Acquisition type	char[32]
Capture Direction	*Ascending or descending*	*char[32]*
Capture Date	*Acquisition date*	*char[11]*
Capture Time	*Acquisition Time*	*char[9]*
Correction Type	*Product type*	*char[5]*
Resample Type	*Resampling method*	*char[4]*
Software Version	*Software version*	*char[11]*
Ingest Software Version	*Ingest software version*	*char[11]*

Table 1 L1G File Metadata Fields

Band Metadata

Field	Description	Type
Band Number	Band Number	integer
Band Name	*LDCM Band designation*	*char[30]*
Upper Left Y	Upper left Y map coordinate	double
Upper Left X	Upper left X map coordinate	double
Upper Right Y	Upper left Y map coordinate	double
Upper Right X	Upper left X map coordinate	double
Lower Left Y	Lower left Y map coordinate	double
Lower Left X	Lower left X map coordinate	double
Lower Right Y	Lower right Y map coordinate	double
Lower Right X	Lower right X map coordinate	double
Projection Distance Y	Y map projection distance	double
Projection Distance X	X map projection distance	double
Maximum Pixel Value	*Maximum DN*	*double*
Minimum Pixel Value	*Minimum DN*	*double*
Pixel Range Valid	*Flag indicating valid pixel max/min*	*integer*
Maximum Radiance	*Maximum radiance*	*double*
Minimum Radiance	*Minimum radiance*	*double*
Spectral Radiance Scaling Offset	*Offset to convert to spectral radiance*	*double*
Spectral Radiance Scaling Gain	*Gain to convert to spectral radiance*	*double*
Radiance valid	*Flag to indicate radiance values are valid*	*integer*
Reflectance Scaling Offset	*Offset to convert to reflectance*	*double*
Reflectance Scaling	*Gain to convert to reflectance*	*double*

Gain		
Reflectance valid	*Flag to indicate radiance values are valid*	*integer*
Instrument Source	*Instrument associated with band imagery*	*char[32]*

Table 2 L1G Band Metadata Fields

7.2.4.8 Maturity

1. Since the OLI 3D grid approach is adopted, the ALIAS code was reused with limited modifications.
2. Due to the detector select option aboard OLI, the detector offset approach has been changed. Under the new methodology the along track detector offsets are stored with the whole pixel adjustment needed due to the detector selected and the small sub-pixel adjustment that was present in the ALI CPF detector offset field. This leads to a need to separate out the fractional detector offset from the detector select offset at times during processing.
3. The problem of multiple terrain intersections needs to be addressed, particularly for off-nadir acquisitions. A terrain occlusion mask will be generated to identify these obstructed pixels (see note #1 below for additional details), but the current thinking is that it would not alter the behavior of the resampler, as sprinkling fill pixels throughout a product image can wreak havoc with some applications. Generating a separate terrain occlusion mask will allow users to evaluate the extent of the problem and apply the mask if appropriate to a particular application. This is being addressed in the Terrain Occlusion ADD.

All items in maturity section have been addressed. The OLI 3D grid approach was adopted. The IPE L1G and L0R libraries were used within the prototype code. The detector delay logic was changed to handle the OLI detector select characteristics. The terrain occlusion ADD addresses the terrain issues associated with the OLI instrument.

7.2.5 Terrain Occlusion Mask Generation Algorithm

7.2.5.1 Background/Introduction

The heritage Landsat and ALI/EO-1 image resampling procedures ignored the possibility of multiple terrain intersections due to off-nadir viewing toward the edges of the imaging swath. This was a reasonable simplification for Landsat with its fixed nadir viewing geometry. Although the ALI was capable of off-nadir pointing, this capability was mostly used to acquire different portions of the nominal Landsat swath, given that the ALI's focal plane was only 20 percent populated. Furthermore, EO-1 was a technology demonstration project with a minimal budget for ground processing algorithm development, so Landsat capabilities were reused as is wherever possible.

Ignoring the multiple terrain intersection effect is less defensible for the pointable OLI which will be acquiring off-nadir scenes from adjacent WRS paths in small, but significant numbers, for product generation. The approach to this problem adopted here is to compute the ground locations where the OLI line of sight is obstructed by terrain, and provide this information in a mask. The image resampling logic will be permitted to populate all output image pixels with apparent values according to the heritage algorithm. Some of these will be erroneous data that actually represent terrain intersection points closer to the imaging sensor. These can be subsequently identified and, if appropriate, replaced with fill by the user based on the contents of the terrain occlusion mask generated by this algorithm. This approach was felt to be preferable to inserting fill in the product image as some image exploitation algorithms (e.g., control point mensuration) are sensitive to the presence of fill.

Generating the terrain occlusion mask can also be performed without reference to the output image itself, requiring only the digital elevation model (registered to the product image output space) and the LOS projection grid as inputs. For each pixel in the output image, the algorithm uses the grid file to locate the corresponding pixel in input (L1R) space. It then uses the grid to compute the output space line/sample location corresponding to the same input line/sample at the maximum elevation plane. The line connecting the original output pixel location with the maximum elevation location corresponds to the projection of that pixel's line-of-sight into output space. By interpolating elevation model heights for points along this line and comparing them to the computed LOS height, terrain intersection points that are closer to the imager can be detected. Each point in the output terrain occlusion mask will be flagged as visible or terrain occlusion.

This is a new algorithm with no ALIAS or Landsat heritage though it will make extensive use of the library functions that access the grid file.

7.2.5.2 Dependencies

The terrain occlusion algorithm assumes that the LOS Projection and Gridding algorithm has created the output product LOS projection grid and that the digital elevation model has been resampled to match the output product frame. The elevation planes in the LOS projection grid must span the range of elevations in the elevation model.

7.2.5.3 Inputs

The terrain occlusion algorithm and its component sub-algorithms use the inputs listed in the following table. Note that some of these "inputs" are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs

ODL file (implementation)
OLI Grid file
DEM Grid file
Original Unresampled DEM file
Terrain Occlusion Mask file name
Terrain Occlusion band

7.2.5.4 Outputs

TO (terrain occlusion) mask file
TO mask data descriptor record (DDR) (see note 4)
TO mask image

7.2.5.5 Options

None.

7.2.5.6 Procedure

Read unresampled DEM to determine maximum elevation within file (maximum_elevation).

Initialize terrain mask to 0.

For each SCA:

For each output pixel:

1) Retrieve the elevation for the current output pixel location (current elevation) from the DEM.
    a. Using DEM resampling grid map L1T output pixel location to geographic unresampled DEM line/sample location.
        i) Calculate grid cell row and column index.

        grid row = output line / number grid cell lines

        grid col = output sample / number grid cell samples

        ii) Determine grid cell number.

        grid cell number = grid row * number grid cell samples + grid col

        iii) Look up grid mapping coefficients based on grid cell.

        coeff = grid cell coefficient reverse[grid cell number].

        iv) Calculate DEM line/sample location.

        DEM line = coeff.line[0] +
        output sample * coeff.line[1] +
        output line * coeff.line[2] +
        output sample * output line * coeff[3].line

        DEM sample = coeff.sample[0] +
        output sample * coeff.sample[1] +

output line * coeff.sample[2] +

output sample * output line * coeff[3].sample

b. Perform bilinear interpolation at location in DEM from step 1a) to determine elevation of current L1T output location.

i) Determine subpixel location

Integer line = (int)DEM line

Integer sample = (int) DEM sample

ds = DEM sample – Integer sample

dl = DEM line – Integer line

ii) Determine location in DEM image buffer.

dem_ns = number samples in DEM

dem_nl = number lines in DEM

loc = Integer line * dem_ns + Integer Sample

iii) Interpolate elevation for floating point location.

elevation =

(1.0 - ds) * (1.0 - dl) * dem.data[loc] +

ds * (1.0 - dl) *dem.data[loc+1] +

(1.0 - ds) *dl * dem.data[loc+dem_ns] +
ds * dl * dem->data[loc+dem_ns + 1]

Note:
For off-nadir images, pixel line-of-sight ground projections can extend outside the product image area. Using the unresampled DEM as the source of elevation data should prevent elevations from being needed outside the available data range as the terrain occlusion calculation performs its “stepping process”. However a check to make sure that the elevation being retrieved is greater than 0 in line and sample while less than dem_nl-1 and dem_ns-1 should be implemented. The process should issue a warning that the data to be retrieved is outside of the DEM, and return the DEM elevation value for the closest edge line/sample position (i.e., clip the DEM line/sample values at the DEM edges).

2) Run ols2ils to find input location for corresponding output location. This will be based on elevation for current output pixels (l_c,s_c).

3) Run get_output_ls for the input location calculated in 2) to find the corresponding output location for the maximum elevation (l_m,s_m).

4) Define the parametric equation for a line that connects (l_c,s_c) to (l_m,s_m).

$s_p = s_0 + t * f$

$l_p = l_0 + t * g$

where: $0 \le t \le 1$

At t=0: $l_p=l_c$ and $s_p=s_c$.

At t=1: $l_p=l_m$ and $s_p=s_m$

Therefore

$l_0 = l_c$,

$s_0 = s_c$,

$g = (l_m - l_c)$,

$f = (s_m - s_c)$

5) Compute the length of the line in output space:

$$d = MAX(1, \sqrt{(s_m - s_c)^2 + (l_m - l_c)^2})$$

6) Compute the increment of t to use to walk along the line:

$$\Delta t = \frac{MAX(|s_m - s_c|, |l_m - l_c|, 1)}{d^2}$$

7) Walk along the line in increments of Δt, testing each point for terrain occlusion:

For j = 0 to (int)ceil(1/Δt)

t = j * Δt

8) Calculate the point of intersection:

$l_p = l_0 + t * g$

$s_p = s_0 + t * f$

9) Round (l_p, s_p) to get (l_p', s_p'). Find the elevation for (l_p', s_p') (pixel elevation) using the DEM resampling grid as described in steps 1a) and 1b) above.

10) The value of t represents the ratio used to measure whether the elevation of (l_p', s_p') is large enough to obscure the current pixel of interest (l_c, s_c).

if((t * maximum elevation + (1.0-t) * current elevation) < pixel elevation)

Current pixel location (l_c, s_c) is occluded. Set terrain mask to 1 and exit loop.

else

Current pixel location (l_c, s_c) is not occluded. Continue to loop.

Determining Elevation (change from using co-registered DEM)

Due to the “walk-a-line” process of step 7) of the previous procedure the location of an elevation requested could reside outside of the co-registered DEM used in creating the L1T. To account for this the unresampled DEM and DEM geomgrid used to resample the DEM can be used to map points from these points outside the L1T geographic extent to that within the unresampled DEM. Since the unresampled DEM should extend outside the boundary of the L1T, this will allow the retrieval of elevations outside the product image extent.

7.2.5.7 Prototype Code

The following is a list of the routines files associated with the prototype code and brief explanation of the purpose of each.

calc_dem_bounds
Takes an image data structure and returns the minimum and maximum values of the data values present. This process defines the boundaries of the searching, or equations, to determine if a pixel has been occluded by another pixel.

getpar
Reads input parameters from an ODL file. Input includes the LOS projection grid, coregistered DEM and the band (or number of bands) to be inspected for pixel occlusion.

occ_get_elevation
Calculates pixel, or elevation, DN from an image data buffer using bilinear interpolation. Input is an IMAGE data structure and floating point location for DN calculation.

occlusion_get_geo
Uses the generic resampling grid to map points from the L1T output location to the unresampled DEM location. Mapping is done through bilinear mapping coefficients stored within the generic grid.

occlusion
Main driver for calculating terrain occlusion mask. Calls getpar to retrieve input parameters, read LOS projection grid, reads DEM file, calls calc_dem_bounds to determine bounds on DEM file, calls terrain_occlusion_mask to calculate mask, frees LOS projection grid from memory, writes occlusion mask to a flat file, and calls write_envi_hdr to create an ENVI header file for occlusion mask.

oli_get_dem
Reads DEM file storing elevation and geographic information into image data structure. Calls several IPE L1G HDF5 routines for reading DEM file.

terrain_occlusion_mask
Module that calculates the terrain occlusion mask. The equations and steps present within this module are listed in the procedure section above. Occlusion mask is calculated using several subroutines present within the terrain_occlusion_mask file:

terrain_occlusion_mask: Main driver for all functions in the terrain_occlusion_mask file. Takes an input of the LOS projection grid, SCA number, elevation data structure, maximum elevation present within coregistered DEM and creates a terrain occlusion mask.

calc_occ_line_eq: Calculates the parametric equations for a line joining two points.

occlusion_build_params: Calculates the length of the line in output space and increment of t parameters.

map_to_input_occlusion: Maps an output space location to an input space location.

calc_occ_scale: Calculates the scale needed to determine if a current pixel is occluded.

write_envi_hdr
Writes out an ENVI header for the occlusion mask.

Prototype dependencies:
1) Input is a HDF4 heritage grid file.

2) DEM file is HDF5 L1G image file.

7.2.5.8 Maturity

4. The problem of multiple terrain intersections needs to be addressed, particularly for off-nadir acquisitions. A terrain occlusion mask will be generated to identify these obstructed pixels (see note #1 below for additional details), but the current thinking is that it would not alter the behavior of the resampler, as sprinkling fill pixels throughout a product image can wreak havoc with some applications. Generating a separate terrain occlusion mask will allow users to evaluate the extent of the problem and apply the mask if appropriate to a particular application.
5. The algorithm does not account for detector specific even/odd and deselect offsets. It generates the mask based on nominal detector locations.
6. The need to have the L1R image available to detect within-image fill conditions (due to nominal detector/band shifting) is overtaken by events, since nominal detector/band alignment fill is not used.
7. Current prototype/test version has only been run on ALI imagery. The processing of generating the mask is currently integrated within the ALIAS resampler code.
8. Early testing with OLI simulated data showed difficulty in defining what portion of a pixel is obstructed, or what portion of another pixel is leading to the obstruction. This may lead to further tweaking of defining the search areas and variables involved in calculating masked pixels, however the under lying principles of the algorithm should remain the same.

7.2.5.9 Notes

Some additional background assumptions and notes include:

1. The new logic required to calculate the terrain occlusion mask (particularly for off-nadir scenes) is documented here, in a separate ADD, but may be implemented as part of the resampling software for processing efficiency. The terrain occlusion (TO) mask output by this algorithm, is also included as a possible (to be resolved) output in the resampling algorithm.
2. The current concept is to allow the user to specify the band(s) to use in testing for occlusion. However for the terrain mask that is to accompany the L1T LDCM product, generation of the mask for the SWIR1 band should be sufficient.
3. Early testing with ALI data showed few pixels being marked as masked. This, along with off-nadir imaging not being a standard product, may lead to changes in how this algorithm will be used during processing and product generation.
4. The DDR will be a component of the output TO mask image file, capturing the metadata necessary to relate mask image pixels to ground positions. This structure is addressed in more detail in the Resampling ADD.

7.2.6 OLI Geometric Accuracy Assessment (L1T)

7.2.6.1 Background/Introduction

The OLI geometric accuracy assessment, or geometric characterization, algorithm analyzes the results of the ground control point (GCP) measurements derived through correlation of the GCP image chips with a precision and terrain corrected OLI L1T image. Unlike the similar geodetic accuracy assessment algorithm, there is no precision LOS model correction step invoked to analyze the GCP results and detect outliers. Instead, the geometric accuracy assessment is applied directly to the results of control point mensuration in the L1T image. Statistics are computed for the GCP measurements, with outliers detected and rejected based upon a t-distribution test. The GCP results provide a measure of the accuracy of the output L1T product through direct comparison to an absolute ground control source. Ideally, a different set of GCPs would be used for geometric accuracy assessment than were used in the L1T LOS model correction process. This will require flagging some GCPs in the GCP library as validation points to be withheld from the original GCP mensuration and LOS model correction process and used only for geometric characterization. Setting these control/validation flags is a Cal/Val Team responsibility.

The OLI geometric accuracy assessment algorithm has no direct ALIAS equivalent, but will be derived from the OLI geodetic accuracy assessment algorithm.

7.2.6.2 Dependencies

The OLI geometric accuracy assessment algorithm assumes that the L1T product generation flow has been executed to create an L1T image, and that this image has been correlated with a set of validation GCP image chips (see note 3) using the GCP correlation algorithm, to produce a set of GCP measurements. Normally this L1T image will be a standard SCA-combined L1T product, but the GCP correlation algorithm may also be run on SCA-separated images in testing and anomaly resolution scenarios.

7.2.6.3 Inputs

The geometric accuracy characterization algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs
ODL file
Measured GCP File Name
Band to process (See note 6)
L1T Image File Name (for access to metadata)
T-distribution outlier threshold
Output geometric report file name
L0Rp ID (for trending)
Work Order ID (for trending)
Characterization Database Output (Trending) Flag (on/off)
GCP residual output (on/off)
Measured GCP File Contents (see GCP Correlation ADD for full details)
GCP point ID
GCP latitude and longitude (in degrees) and height (in meters)
Correlation success/failure flag
Predicted GCP image line and sample location

Line and sample offsets in pixels
Measured band number
GCP source (GLS or DOQ)
L1T Image File Metadata Contents (see Image Resampling ADD for full details)
WRS Path/Row (for trending)
Image pixel size (for measured band) in meters
Acquisition date (for trending)
Acquisition type (for standard report file header)
Scene roll angle (for trending and report file)

7.2.6.4 Outputs

Geometric Accuracy Report (output file and trending) (see Table 1 below for additional details)
Processing Information
Processing Date and Time
Processing Center/Location
Processing Software Version
Processed L1T Image File Name
Data Set Information
Spacecraft and Instrument Source (LDCM/OLI)
Work Order ID
WRS Path/Row
Off-Nadir Angle
Acquisition Type (Earth, Lunar, Stellar) (will always be Earth)
L0Rp ID
Acquisition Date
GCP Information
GCP Source
Pixel Size (meters)
Number of valid GCPs
Mean latitude and longitude of GCPs
GCP Statistics
Mean, RMSE, Standard Deviation, Correlation Coefficient
GCP Residual Output Fields (written to report file only if option is selected)
For each valid GCP:
GCP point ID
GCP latitude and longitude (in degrees) and height (in meters)
Line and sample offsets scaled to meters
Measured band number
Measured SCA number (usually 0) (see note #4)

7.2.6.5 Options

Characterization database output on or off.
GCP residual output on or off.

7.2.6.6 Procedure

Geometric characterization is performed on the measured GCP file output created by the GCP correlation algorithm. See the GCP Correlation Algorithm Description Document (ADD) for further details on the GCP mensuration process and its results. Geometric characterization reads the entire set of GCP measurements, removes those flagged as correlation failures, performs a Student t-

distribution test to detect and eliminate outliers, and calculates statistics for the remaining valid residuals. The resulting statistics reflect the accuracy of the measured L1T image product relative to the set of validation GCPs used in the correlation process, providing a measure of L1T product accuracy that is independent of the control used to create the product.

The geometric accuracy assessment algorithm computes the same summary statistics as the geodetic accuracy assessment algorithm. It differs from that algorithm in that it must read the raw GCP measurement file produced by the GCP correlation algorithm instead of the precision correction residuals file produced by the LOS model correction algorithm and therefore must filter its input for outliers prior to computing the statistics. Also unlike geodetic characterization, geometric characterization includes an option to write out the measured offsets for those GCP declared as valid by the outlier test. This makes it possible to capture the outlier test results for individual GCPs for further analysis.

Figure 1 shows the architecture for the Geometric Characterization algorithm.

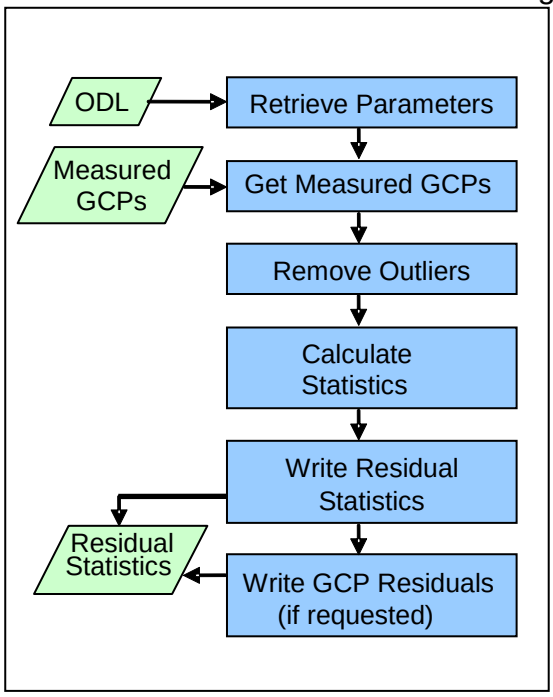


Figure 1: Geometric Characterization Algorithm Architecture

The geometric accuracy assessment algorithm consists of three stages:

Stage 1 - read processing parameters and load GCP data.
Stage 2 - use the Student t-distribution test to perform outlier filtering on the GCPs.
Stage 3 - compute the mean, standard deviation, RMSE, and correlation coefficient statistics for the valid GCPs, and generate the report file and characterization database outputs.

In practice, stages 2 and 3 are somewhat overlapped as it is necessary to compute mean and standard deviation statistics on the GCP set as part of the t-distribution outlier test.

Stage 1 - Load Processing Parameters and GCP Measurements

Stage 1 processing reads the processing parameters from the input ODL file and loads the GCP data from the measured GCP file.

Get Geometric Characterization Parameters Sub-Algorithm (new)
Read the parameters for geometric characterization from the input ODL parameter file. Also read the Level 1T image metadata to retrieve the WRS path/row, acquisition date, and pixel size for all bands in the image.

Get GCP Measurements Sub-Algorithm
This function reads the latitude, longitude, band number, GCP source, and correlation status flags as well as the along- and across-track offset measurements for each GCP from the measurement file created by the GCP correlation algorithm. It retrieves all of the measured GCPs for subsequent correlation flag and outlier filtering. Each GCP record is loaded into a data structure containing the fields listed in Table 1 of the prototype section.:

The measured GCP line and sample offsets are scaled to units of meters using the pixel size read from the L1T metadata for the measured band. The fields required for stage 2 and 3 processing are:

1. Point ID
2. GCP Latitude
3. GCP Longitude
4. GCP Height
5. Line Offset Scaled to Meters
6. Sample Offset Scaled to Meters
7. Valid GCP (Correlation and Outlier) Flag
8. L1T Band Number
9. L1T SCA Number (usually 0 for SCA-combined images) (see note #4)
10. GCP Source

Stage 2 - Filter GCP Outliers
Stage 2 processing identifies those GCP measurements classified as outliers. The outlier set is initially identified as those GCPs flagged as image correlation failures. A statistical t-distribution outlier test is applied iteratively to the remaining valid GCPs, removing any newly identified outliers at the end of each iteration, until no new outliers are found.

Remove GCP Outliers Sub-Algorithm (new)
This function removes the GCP records flagged as outliers from the valid GCP set. Records with the correlation flag field set to 0 are outliers. The initial valid GCP set are those GCPs that were flagged as successful correlations. The Student t-distribution outlier test is then performed to identify additional outlier GCPs based on the magnitude of their offsets relative to the mean offsets for the entire valid set.

Student-T Outlier Test Sub-Algorithm
Given a tolerance value, outliers are removed within the data set until all values deemed as “non-outliers” or “valid” fall inside the confidence interval of a T-distribution. The tolerance, or probability threshold for the associated confidence interval, is specified per run and usually lies between 0.9-0.99. The default value is 0.95. The number of degrees of freedom of the data set is equal to the number of valid data points minus one. The steps involved in this outlier procedure are as follows:

1) Calculate mean and standard deviation of valid GCP data set for both the line and sample directions.

2) Find largest offset and compare it to outlier threshold.

a) Find two tailed T-distribution (*T*) value for current degree of freedom and confidence level specified α. See the T-Distribution Confidence Interval Computation discussion below for details on how the T-distribution confidence interval limit value (*T*) is computed based on the specified probability threshold (α). Note that this must be recomputed for each iteration since the number of degrees of freedom changes as outliers are removed.

b) Calculate largest deviation from the mean allowable for the specified degree of freedom and α:

$\Delta line = \sigma_{line} * T$

$\Delta sample = \sigma_{sample} * T$

Where:

σ_{line} = standard deviation of valid line offsets

σ_{sample} = standard deviation of valid sample offsets

c) Find valid data point that is farthest from the mean.

$max\ line_i$ = MAX{ line offset - mean line offset}

$max\ sample_j$ = MAX{ sample offset - mean sample offset}

Where:

The maximum is found from all valid offsets

i is the tie-point number of max line

j is the tie-point number of max sample

d) If valid data point that is farthest from the mean is greater than the allowable Δ then the valid point is flagged as outlier.

if $max\ line_i > \Delta line$ or $max\ sample_j > \Delta sample$ then

if($max\ sample_j / \sigma_{sample} > max\ line_i / \sigma_{line}$)

tie-point j is marked as an outlier

else

tie-point i is marked as an outlier

else no outliers found

3) Repeat 1 and 2 above until no outliers are found.

T-Distribution Confidence Interval Computation

The probability density function (pdf) for the t-distribution is:

$$f_r(t) = \frac{\Gamma(\frac{r+1}{2})}{\sqrt{\pi r}\,\Gamma(\frac{r}{2})\left(1+\frac{t^2}{r}\right)^{(r+1)/2}}$$

where: r = the number of degrees of freedom (n-1)

Γ = the gamma function

As pointed out in "Numerical Recipes in C", it is often more convenient (and safer) to compute the logarithm of the gamma function as the gamma function values can get quite large and it is often ratios of gamma functions (as here) that are of interest. The t-distribution pdf can be reformulated as:

$$f_r(t) = e^{u(t)}$$

$$u(t) = \ln\Gamma(\frac{r+1}{2}) - \frac{\ln(\pi r)}{2} - \ln\Gamma(\frac{r}{2}) - \frac{r+1}{2}\ln\left(1+\frac{t^2}{r}\right)$$

where: $\ln\Gamma$ = the logarithm of the gamma function

"Numerical Recipes in C" provides a routine for computing $\ln\Gamma$ called gammln (ref. page 214 of the 2nd edition). The gammln function is used in the reformulated t-distribution pdf shown above to compute the value of the pdf for a given t and degrees of freedom (r).

Using the t-distribution pdf we compute the confidence level by numerical integration:

1. Initialize the integration, setting the integration step size to 0.001:
    a. step = 0.001
    b. sum = 0
    c. target = α/2 (half the probability threshold since we're only integrating the positive half of the distribution)
    d. t = 0
    e. delta = step*(t_pdf(t,dof) + t_pdf(t+step,dof))/2
2. Iterate the integration steps until the sum reaches the target:
    while(sum+delta < target)
    a. sum = sum + delta
    b. t = t + step
    c. delta = step*(t_pdf(t,dof) + t_pdf(t+step,dof))/2
3. Solve for the Δt value to exactly reach target, assuming the pdf is linear over the step size:
    a. a = (t_pdf(t+step,dof) - t_pdf(t,dof))/2/step
    b. b = t_pdf(t,dof)
    c. c = sum - target
    d. if (|a| > 0) Δt = (-b + sqrt(b^2 - 4ac))/2/a (the quadratic formula)
    else if (|b| > 0) Δt = -c/b
    else Δt = 0
4. Compute the final T value: T = t + Δt

Stage 3 - Calculate GCP Statistics and Create Output

The third stage of processing calculates the summary statistics for the final valid GCP set and generates the output geometric accuracy report and characterization database outputs.

Analyze GCP Residuals Sub-Algorithm

This function calculates the mean, root mean square error (RMSE), and standard deviation of the along- and across-track GCP residuals as well as the correlation coefficient between the across- and along-track residuals. The statistics are computed in the following process:

a) Calculate GCP statistics
    a1) Calculate total number of GCPs used (count of valid GCPs)
    a2) Calculate mean latitude of GCPs used
    a3) Calculate mean longitude of GCPs used

b) Calculate offset statistics
    b1) Calculate mean of line offsets

b2) Calculate mean of sample offsets
b3) Calculate RMSE of line offsets
b4) Calculate RMSE of sample offsets
b5) Calculate standard deviation of line offsets
b6) Calculate standard deviation of sample offsets
b7) Calculate correlation coefficient between line and sample offsets

The following equations are used to perform these calculations, with X being the parameter for which statistics are calculated:

Mean:

$$X_{mean} = \frac{1}{numGCP} \sum_{i=1}^{numGCP} X_i$$

RMSE:

$$X_{RMS} = \sqrt{\frac{1}{numGCP} \sum_{i=1}^{numGCP} X_i^{\,2}}$$

Standard Deviation:

$$X_{StdDev} = \sqrt{\frac{1}{numGCP-1}\left(\left(\sum_{i=1}^{numGCP} X_i^{\,2}\right) - numGCP * X_{mean}^{\,2}\right)}$$

Correlation Coefficient:

$$XY_{CC} = \frac{\left(\sum_{i=1}^{numGCP}\left(X_i - X_{mean}\right)\left(Y_i - Y_{mean}\right)\right)}{\left(numGCP-1\right)} \frac{1}{X_{StdDev} Y_{StdDev}}$$

Output Geometric Statistics Sub-Algorithm
This function creates the output geometric report file and writes the statistics computed from the GCP offsets to the output file. Note that the output of trending data to the characterization database is performed by the geometric characterization main procedure.

Write Geometric Statistics Sub-Algorithm
This function writes the standard report file header and then writes the GCP offset statistics to the ASCII output file.

Write GCP Residuals Sub-Algorithm
If the option to write the valid GCP measurements to the output report file is selected, this sub-algorithm is invoked to loop through the GCP list, writing those with the outlier/valid flag set to 1 to the output report file. The individual GCP measurements are not written to the characterization database. The following fields are written:

1. Point ID
2. GCP Latitude (in degrees)
3. GCP Longitude (in degrees)
4. GCP Height (in meters)
5. Line Offset (scaled to meters)

6. Sample Offset (scaled to meters)
7. L1T Band Number
8. L1T SCA Number (will be 0 for SCA-combined images) (see note #4)

Algorithm Output Details

The geometric accuracy assessment algorithm outputs are summarized in Table 5 below. All fields are written to the output report file (subject to the GCP residual output flag setting) but only those with "Yes" in the "Database Output" column are written to the characterization database. Note that the first eleven fields listed constitute the standard report header.

Field	Description	Database Output
Date and time	Date (day of week, month, day of month, year) and time of file creation.	Yes
Spacecraft and instrument source	LDCM and OLI	Yes
Processing Center	EROS Data Center SVT (see note #5)	Yes
Work order ID	Work order ID associated with processing (blank if not applicable)	Yes
WRS path	WRS path number	Yes
WRS row	WRS row number	Yes
Software version	Software version used to create report	Yes
Off-nadir angle	Scene off-nadir roll angle (in degrees)	Yes
Acquisition type	Earth, Lunar, or Stellar (only Earth-viewing scenes are used for geometric characterization)	Yes
L0Rp ID	Input L0Rp image ID	Yes
L1T image file	Name of L1T used to measure GCPs	No
Acquisition date	Date of L1T image acquisition	Yes
GCP source	Source of GCPs (GLS or DOQ)	Yes
Pixel size	L1T image pixel size (for measured band) in meters	Yes
Number of valid points	Number of GCPs accepted as valid	Yes
Mean GCP latitude	Mean latitude of valid GCPs (degrees)	Yes
Mean GCP longitude	Mean longitude of valid GCPs (degrees)	Yes
Line offset mean	Mean of line offsets scaled to meters	Yes
Sample offset mean	Mean of sample offsets scaled to meters	Yes
Line offset RMSE	RMSE of line offsets scaled to meters	Yes
Sample offset RMSE	RMSE of sample offsets scaled to meters	Yes
Line offset standard deviation	Standard deviation of line offsets scaled to meters	Yes

Sample offset standard deviation	Standard deviation of sample offsets scaled to meters	Yes
Correlation coefficient	Correlation coefficient between line and sample offsets (dimensionless)	Yes
If the residual output option is selected:	For each valid GCP:	
Point ID	GCP ID (see GCP Correlation ADD for format)	No
GCP Latitude	GCP WGS84 latitude in degrees	No
GCP Longitude	GCP WGS84 longitude in degrees	No
GCP Height	GCP WGS84 ellipsoid height in meters	No
Line Offset	Measured line offset scaled to meters	No
Sample Offset	Measured sample offset scaled to meters	No
L1T Band Number	Band in which GCP was measured	No
L1T SCA Number	SCA in which GCP was measured (0 for SCA-combined images) (see note #4)	No

Table 5: Geometric Accuracy Assessment Output Details

Accessing the Geometric Accuracy Characterization Database

Though not part of the formal geometric accuracy assessment algorithm, some comments regarding the anticipated methods of accessing and analyzing the geometric accuracy results stored in the characterization database may assist with the design of the characterization database.

The database output from the geometric accuracy assessment algorithm will be accessed by a statistical summary analysis tool that queries the characterization database to retrieve geometric accuracy results from multiple scenes. Summary mean and RMSE statistics for the scene results will be calculated and output in a report containing a comma-delimited table of the retrieved trending results as well as the summary statistics.

The geometric results would typically be queried by acquisition date, scene off-nadir angle, WRS path/row, and/or GCP source. The most common query would be a combination of GCP source, scene off-nadir angle, and acquisition date range, for example, selecting all of the GLS-derived results, from nadir scenes, for a given calendar quarter:

GCP_Source = "GLS"
Off_Nadir_Angle is between -0.5 and 0.5
Acquisition_Date is between 01APR2012 and 30JUN2012

The summary mean and RMSE statistics would be calculated from the mean and RMSE results for the individual scenes returned as:

$$Mean_{net} = \frac{1}{numScene} \sum_{i=1}^{numScene} Mean_i$$

$$RMSE_{net} = \sqrt{\sum_{i=1}^{numScene} RMSE_i^{\,2} \Big/ numScene}$$

The query results would be formatted in a set of comma-delimited records (for ease of ingest into Microsoft Excel), one record per scene. Each record would contain all of the fields written to the characterization database (items with "Yes" in the rightmost column of Table 5 above). A header row containing the field names should precede the database records. Two trailer rows, one containing the summary statistic names (Net Line Offset Mean, Net Sample Offset Mean, Net Line Offset RMSE, Net Sample Offset RMSE) and the second containing the comma-delimited summary statistic values, should follow the database records.

7.2.6.7 Prototype Code

The correlation or mensuration process for the Geometric Accuracy Assessment ADD is the same as that listed in the GCP Correlate ADD. The GCP Correlate ADD should be referenced for both the procedure and prototype for that portion of the Geometric Accuracy Assessment algorithm. The outlier rejection prototype is the only portion of the Geometric Accuracy Assessment ADD that is discussed in this document.

Input to the correlation and outlier rejection executables are ODL files. These ODL files, called geometric.odl and tdist.odl, are listed in the test data directory. Output from the correlation process (gcpcorrelate) is an ASCII file containing measured offsets between the search and reference data files. Output from the outlier rejection process (tdist) is several files; one is a reformatted version of the input file, a second is a file containing the correlation points flagged as outliers, a third is a set of statistics calculated on the valid correlation points, a fourth is a set matching the output listed in table 5 of this ADD. Also under the test directory is a perl script that will reformat the output created from gcpcorrelate into the expected format of tdist. The tdist function is also used by the Image Accuracy Assessment and Band Accuracy Assessment ADDs. These processes have a slightly different output file format than the gcpcorrelate process, thus the need to reformat the gcpcorrelate output file using the perl script.

The prototype code was compiled with the following options when creating the test data files:

```
-g -Wall -march=nocona -m32
```

Main driver (tdist)
Driver for outlier rejection process. Calls functions to read ODL parameters (getpar), read unfiltered, and reformatted, gcpcorrelate output (get_gcpdata), loops on SCAs and band combinations filtering outliers, writes output files (put_gcpdata, put_resdata, put_gcpstats). These output files include a reformatted gcpcorrelate output file, a reformatted gcpcorrelate output file with outliers flagged, and a statistics file of the final filtered results.

Get input parameters (getpar)
Parses ODL file for the following input parameters; t-student outlier tolerance, residuals output file name, input file name (reformatted output from gcpcorrelate), the output statistics file name, a switch for combining statistics of a SCA separated image file, and a switch for printing valid individual points within one of the requested output files.

Read input data file (get_gcpdata)
Reads either an Image Accuracy Assessment or Band Accuracy Assessment formatted file. These files contain the correlation results between a given set of source or image files. All relevant information needed for assessment of the correlation results is stored within the GCPDATA structure.

Filter outlier (filter_outliers)
Performs a student-t outlier rejection on a given set of correlation points.

Calculate the confidence level for a two-tailed t-distribution (xx_t_conf)
Calculates and returns the threshold value corresponding to the confidence level of a two-tailed t-distribution with a specified number of degrees of freedom. Note that this function replaced the historical t_conf module that was present in the ALIAS tdist code.

Calculate the value of a t-distribution probability density function (PDF) at a given point (t_pdf-contained within file xxx_t_conf)
Needed for calculating the confidence level of the student-t test.

Calculate the natural log the gamma function (gammln - contained within file xxx_t_conf)
Needed for calculating the confidence level of the student-t test. Specifically, used for estimating the PDF of the t-distribution.

Write out reformatted input file (put_gcpdata)
Creates file of original gcpcorrelate results only in a reformatted output context. Outliers are not flagged.

Write out reformatted input file with outliers' flagged (put_resdata)
Creates file of original gcpcorrelate results only in a reformatted output context. Outliers are flagged.

Write out statistics file (put_gcpstats)
Creates a statistics file of filtered correlation results. Results are calculated for both the line and sample directions; mean, maximum, minimum, average, median, standard deviation, and root mean square (RMS) error. Median is calculated through compute_median, all other statistics are calculated within put_gcpstats. Statistics are split by both band combination and SCA if necessary.

Write out accuracy assessment file (put_geostats)
Creates a geometric accuracy assessment file. Calls library functions to calculate mean, standard deviation, root mean error, and correlation coefficient. Will also print out individual valid GCPs, or correlation locations, if requested.

Compute median of filtered correlation results (compute_median)
Computes median of filtered correlation results. Values are sorted using a heap sort method, the middle element of heap sort is chosen as the median value. A value for both the line and sample direction is calculated.

Create LDCM "like" output header (output_header.c)
Creates basic LDCM output header information within a file.

Sort correlation results (heapsort)
Sort a given set of values using a heap sort methodology.

Discussion on prototype, and final, report file formats
As discussed above, the prototype for Geometric Accuracy Assessment creates five output files during processing. This level of reporting is not necessary for final implementation. The file created from tdist that contains the same information as the input and contains the .out extension is not

necessary for processing. The final statistics listed in the Algorithm Output sections below (Table 1) shows the reporting and trending that is needed for the Geometric Accuracy Assessment.

The following tables lists the output file formats from the Geometric Accuracy Assessment prototype:

The output from gcpcorrelate processing, containing original unfiltered correlation results, is listed below. This file is called measured.gcp in the test data directory.

Field	**Description**
GCP Record Fields:	One set per GCP
Point ID	GCP ID
GCP chip line location	Line location of GCP within chip
GCP chip sample location	Sample location of GCP within chip
GCP latitude	GCP WGS84 latitude in degrees
GCP longitude	GCP WGS84 longitude in degrees
GCP height	GCP WGS84 ellipsoid height in meters
Predicted GCP image line	Predicted line location of GCP in L1G image
Predicted GCP image sample	Predicted sample location of GCP in L1G image
GCP image line offset	Measured line offset from predicted location
GCP image sample offset	Measured sample offset from predicted location
Correlation success flag	Flag 0 = correlation failure, 1 = success
Correlation coefficient	Measured correlation coefficient (new)
Search band number	L1G band number used
Search SCA number	L1G SCA where GCP was found
Chip source	GCP source (DOQ or GLS or TM6)

Table 1. Output From GCPCorrelate

The output format created from the perl script meas2char2.pl is listed below. This file is a reformatting of the gcpcorrelate output file, reformatted to match the output from the Image Accuracy Assessment ADD. This format is one of the two acceptable formats to the tdist function. This file has been referenced as the Data File (.dat extension) in the Image Accuracy Assessment ADD and is called geometric.dat in the test data directory.

Field	**Description**
Date and time	*Date (day of week, month, day of month, year) and time of file creation.*
Spacecraft and instrument source	*LDCM*
Processing System	*IAS*
Work order ID	*Work order ID associated with processing (blank if not applicable)*
WRS path/row	*WRS path and row*
Software version	*Software version used to create report*
L0R image file	*L0R image file name used to create L1T*
Processed image file name	*Name of L1T used to create report*

Reference bands	Reference bands used in image assessment
Search bands	Search bands used in image assessment
Heading for individual tie-points	One line of ASCII text defining individual tie-point fields.
For each tie-point:	
Tie point number	Tie-point index/number in total tie-point list
Reference line	Tie-point line location in reference image (band)
Reference sample	Tie-point sample location in reference image (band)
Reference latitude	Tie-point latitude location
Reference longitude	Tie-point longitude location
Reference elevation	Elevation of tie-point location
Search line	Tie-point line location in search image
Search sample	Tie-point sample location in search image
Delta line	Measured offset in line direction
Delta sample	Measured offset in sample direction
Outlier flag	1=Valid, 0=Outlier
Correlation Coefficient	Correlation peak coefficient from correlation process
Reference band	Reference band number
Search band	Search band number
Reference SCA	SCA number that reference window was extracted from
Search SCA	SCA number that search window was extracted from
Search image	Name of search image
Reference image	Name of reference image

Table 2. Data File Output from TDIST

The file format for the correlation results with outliers flagged is listed in the table below. This file has been referenced as the Residuals File (.res extension) in the Image Accuracy Assessment ADD. This file is created from the tdist executable. There is also a file with the same name as this Residuals File with an .out extension which is also created from the tdist process. This file has the same format as the Residuals File but does not have the outliers flagged. These files are called geometric.res and geometric.out respectively in the test data directory. These files will have the format as that listed below.

Field	**Description**
Date and time	*Date (day of week, month, day of month, year) and time of file creation.*
Spacecraft and instrument source	*LDCM*
Processing System	*IAS*
Work order ID	*Work order ID associated with processing (blank if not applicable)*
WRS path/row	*WRS path and row*
Software version	*Software version used to create report*
L0R image file	*L0R image file name used to create L1T*
Processed image file name	*Name of L1T used to create report*
Number of records	Total number of tie-points stored in file

Heading for individual tie-points	One line of ASCII text defining individual tie-point fields.
For each band combination	
Combination header	Number of points in combination, reference band number, search band number.
For each tie-point:	
Tie point number	*Tie-point index/number in total tie-point list*
Reference line	Tie-point line location in reference image (band)
Reference sample	Tie-point sample location in reference image (band)
Reference latitude	Tie-point latitude location
Reference longitude	Tie-point longitude location
Reference elevation	Elevation of tie-point location
Search line	Tie-point line location in search image
Search sample	Tie-point sample location in search image
Delta line	*Measured offset in line direction*
Delta sample	*Measured offset in sample direction*
Outlier flag	1=Valid, 0=Outlier
Reference band	Reference band number
Search band	*Search band number*
Reference SCA	SCA number that reference window was extracted from
Search SCA	*SCA number that search window was extracted from*
Search image	Name of search image
Reference image	Name of reference image

Table 3. Filtered Data Points From TDIST

The format of the statistics file for the filtered correlation results are listed below. This file has been referenced as the Statistics File (.stat extension) in the Image Accuracy Assessment ADD. This file is created from the tdist executable and is called geometric.stat in the test directory.

Field	**Description**
Date and time	*Date (day of week, month, day of month, year) and time of file creation.*
Spacecraft and instrument source	*LDCM*
Processing System	*IAS*
Work order ID	*Work order ID associated with processing (blank if not applicable)*
WRS path/row	*WRS path and row*
Software version	*Software version used to create report*
L0R image file	*L0R image file name used to create L1T*
Processed image file name	*Name of L1T used to create report*
t-distribution threshold	*Threshold used in t-distribution outlier rejection*
For each band combination	
Reference band	Reference band of statistics

Search band	Search band of statistics
SCA	SCA number of search image
Total tie-points	Total number of tie-points for band
Correlated tie-points	Number of tie-points that successfully correlated for band
Valid tie-points	*Total number of valid tie-points for band after all outlier rejection has been performed*
For both line and sample direction:	All statistics are given in terms of pixels
Minimum offset	Minimum offset within all valid offsets
Mean offset	*Mean offset of all valid offsets*
Maximum offset	Maximum offset within all valid offsets
Median offset	Median offset within all valid offsets
Standard deviation	*Standard deviation of all valid offsets*
Root-mean-squared	*Root mean squared offset of all valid offsets*

Table 4. Statistics File From TDIST

The final output is the Geometric Accuracy Assessment file and follows the format for the Geometric Accuracy Assessment Algorithm shown in table 5 of the Algorithm Output Details section. Most of these fields are drawn from items also stored with table 1-4 above and are italicized within the tables themselves. This redundancy is from historical aspects of the original code that the prototype code was drawn from.

7.2.6.8 Maturity

The main differences between the geodetic characterization and geometric characterization algorithms are:

1. Different input data formats (precision residual file vs. GCP measurement file).
2. Only one set of GCP measurements to analyze for geometric assessment.
3. Need to detect and reject outliers for geometric assessment.

As with the geodetic characterization algorithm, a field to capture the GCP source (GLS vs. DOQ) has been added.

7.2.6.9 Notes

Some additional background assumptions and notes include:

1. The RMSE GCP statistics capture the absolute geometric accuracy performance of the LDCM output L1T product.
2. The trending output from this algorithm will be accessed by a statistical summary analysis tool that queries the trending database to retrieve geometric accuracy results from multiple scenes. Summary statistics (mean, standard deviation, and RMSE) for the individual scene results will be calculated and output in a report containing a comma-delimited table of the retrieved trending results as well as the summary statistics.
3. The GCPs in the GCP repository (part of the Infrastructure Element) will be flagged as either “control” points, to be used for LOS model correction, or “validation” points, to be used for geometric accuracy assessment. Either the utility that extracts control points from this repository or the GCP correlation algorithm will extract the desired GCP set. In either case, geometric accuracy assessment would operate on the resulting output from GCP correlation.

The “control” set would contain the majority of the points. The “validation” flag would only be used in areas where more than some minimum threshold number of GCPs are available. These flags would be set by the CalVal Team at the time the GCP repository was loaded and could be adjusted, if necessary, thereafter.

4. The GCP residual output option includes writing the GCP SCA number to the output report file. Under normal conditions this field will always be zero, indicating that GCP mensuration was performed on an SCA-combined image. For anomaly investigation and testing purposes it may be desirable to perform GCP mensuration on an SCA-separated image. For example, to use geometric accuracy assessment to analyze the GCP correlation output for a scene that failed LOS model creation for no immediately obvious reason. Thus, support for tracking the SCA where GCPs were measured is retained in this algorithm.
5. A configuration table (system table) should be provided for each installation of the algorithm implementation to convey site-specific information such as the processing center name (used in the standard report header), the number of processors available (for parallel processing implementations), etc. This takes the place of the heritage system table which also contained certain algorithm-related parameters. Anything related to the algorithms has been moved to the CPF for LDCM.
6. It is worth noting that the band to process was added to the input table of Geometric Accuracy Assessment v3.0 ADD. The comment was made during discussions of the GCP Correlate v3.0 ADD to add this as an option to the input parameters. The prototype, for GCP Correlate and Geometric Accuracy Assessment, will default to the first band present within the image file as the band to process if the PAN band is not present. Adding the band number as an input makes the application more robust and can be added during the implementation phase of the algorithm.

7.2.7 OLI Geodetic Accuracy Assessment (L1Gs)

7.2.7.1 Background/Introduction

The OLI geodetic accuracy assessment, or geodetic characterization, algorithm analyzes the results of the ground control point (GCP) measurements created by the LOS model correction algorithm to assess the geolocation accuracy of the systematically terrain corrected OLI L1G image used for GCP mensuration. Statistics are computed for the original (unadjusted) GCP measurements and for the final (best fit) adjusted GCP locations. In both cases, GCPs identified as outliers by the LOS model correction algorithm are excluded. The “pre-fit” results, based upon the unadjusted GCP measurements, provide a measure of LDCM pointing, position, and alignment knowledge as reflected in the measured geolocation accuracy. The “post-fit” results, based on the control point residuals after the precision LOS model corrections are applied, provides a measure of how well the precision correction process is working and an indication of the quality of the derived model corrections. This is particularly important in the LDCM environment in which precision correction using GCPs will be attempted on all scenes, even those with cloud cover, and it will be necessary to identify those cases where the control point matching and precision correction process has failed. The LOS model correction algorithm will perform these tests operationally and the geodetic accuracy assessment algorithm will not be executed for scenes where the precision correction process is known to have failed. The geodetic accuracy assessment results will help identify cases where a substandard precision correction solution has been accepted, thereby assisting in tuning the parameters used to detect precision correction failures.

Geodetic accuracy assessment will be performed as part of the processing flow for the standard L1T scenes, processed using GCPs extracted from the Landsat Global Land Survey (GLS) data. It will also be performed during calibration site processing, using the more accurate GCPs derived from digital orthophoto quad (DOQ) control. Though the geodetic accuracy assessment process is the same for both of these uses, the results will be trended separately as the GLS results will be used to assess the quality of the GLS global control as well as the accuracy of the LDCM products (see note #2), whereas the high-accuracy DOQ control will be used to assess the performance of the operational LDCM navigation and geometric calibration (see note #1).

The OLI geodetic accuracy assessment algorithm is derived from the ALI geodetic characterization algorithm which was, in turn, derived from the corresponding Landsat 7 algorithm. Since this algorithm primarily involves computing statistics on control point measurements, the logic is somewhat sensor independent.

7.2.7.2 Dependencies

The OLI geodetic accuracy assessment algorithm assumes that the LOS model correction algorithm, and its predecessors, has been executed to create an output GCP residuals file. This file is parsed by the geodetic accuracy algorithm to extract the residuals for all non-outlier GCPs for both the first (unadjusted) and last (final) iterations of the LOS model correction process. Note that these residuals are recorded as along- and across-track offsets by the LOS model correction algorithm.

7.2.7.3 Inputs

The geodetic characterization algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs
ODL file

Input residual file name
Output geodetic report file name
Level 1G mensuration image file name
L0R ID (for trending)
Work Order ID (for trending)
Trending flag (on/off)
Level 1G Image File Contents (see note #5 and Resampling ADD for details)
WRS Path/Row (for trending) from image metadata/DDR
Scene acquisition date (for trending) from image metadata/DDR
Scene acquisition type (for trending) from image metadata/DDR
Scene roll angle (for trending/report file) from image metadata/DDR
Residual File Contents (see LOS Model Correction ADD for details)
GCP Latitude/longitude/height
GCP outlier/valid flag
Cross-track and along-track pre-fit residuals
Cross-track and along-track post-fit residuals
GCP source (GLS or DOQ) (new)

7.2.7.4 Outputs

Geodetic Accuracy Report (output file and trending) (see Table 1 below for additional details)
Processing Information
Processing Date and Time
Processing Center/Location
Processing Software Version
Processed L1G Image File Name
Data Set Information
Spacecraft and Instrument Source (LDCM/OLI)
Work Order ID
WRS Path/Row
Roll Angle (new)
Acquisition Type (Earth, Lunar, Stellar) (will always be Earth)
L0Rp ID
Acquisition Date (new)
GCP Information
GCP Source (new)
Number of valid GCPs
Mean latitude and longitude of GCPs
Pre-Fit Statistics
Mean, RMSE, Standard Deviation, Correlation Coefficient
Post-Fit Statistics
Mean, RMSE, Standard Deviation, Correlation Coefficient

7.2.7.5 Options

Trending on or off.

7.2.7.6 Procedure

Geodetic characterization is performed on the precision residual file. See the LOS Model Correction Algorithm Description Document (ADD) for further details on the correction process and its results. Geodetic characterization calculates statistics for the post and pre-fit residuals of the precision correction process. This process allows analysis of the accuracy of LOS model and its performance

when processing image data using only the spacecraft ancillary data, as compared to applying ground control.

Since the LOS model correction algorithm detects and flags outlier GCPs, the work of the geodetic accuracy assessment algorithm is limited to reading and parsing the output residual file created by LOS mode correction, and computing summary statistics for the results of the first and last solution iterations. These results are written to a report file along with standard header fields, some of which are extracted from the L1G image metadata. Figure 1 shows the architecture for the geodetic characterization algorithm.

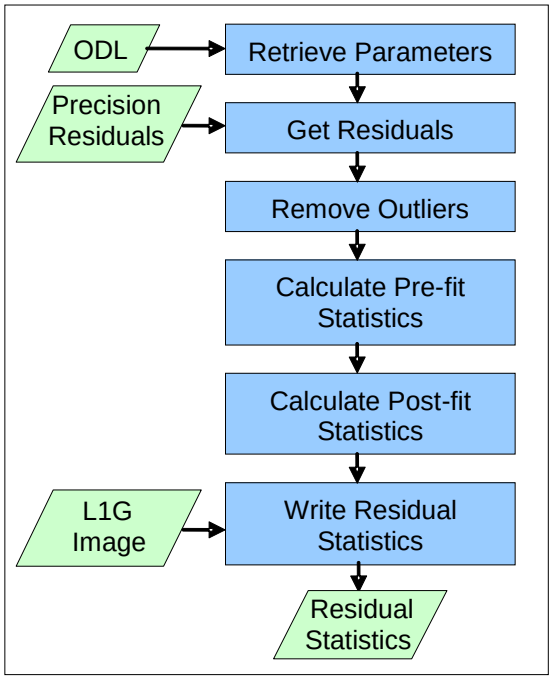


Figure 1: Geodetic Characterization Algorithm Architecture

7.2.7.7 Prototype Code

Inputs to the executable are an ODL parameter file, an ASCII residual file generated by the LOS correction algorithm, and the L1G image used to measure the GCPs. Note that the L1G image is only used to provide metadata for inclusion in the output report. The output is an ASCII report file containing a standard header that identifies the data set analyzed, and the pre-fit and post-fit GCP residual summary statistics for the GCPs used in the precision correction solution. The prototype code also accesses two environment variables to populate fields used in the standard report header. These are IAS_REL which contains the IAS software version number, and IAS_SITE which contains a text string identifying the processing center.

The prototype code was compiled with the following options when creating the test data files:

```
-g -Wall -march=nocona -m32
```

Get Geodetic Characterization Parameters Sub-Algorithm (get_geod_char_parms)
This function gets the parameters for geodetic characterization from the input ODL parameter file. It also reads the Level 1G image metadata to retrieve the WRS path/row, acquisition date, acquisition type, and scene roll angle.

Read Grid Parameters Sub-Algorithm (read_grid_parm)

This function reads the standard parameters common to all applications from the input ODL parameter file. These include the script name, Level 0R ID, work order ID, and trending on/off flag.

Get Residual Sub-Algorithm (xxx_get_residual)

This function reads the along and across-track residual components for each GCP from the residual file created by the LOS model correction algorithm. It retrieves all of the residuals for a specified iteration. If a negative iteration number is provided it retrieves the data for the final iteration. This sub-algorithm is invoked twice, once to retrieve the residuals for iteration 0 (pre-fit) and once to retrieve the final iteration residuals (post-fit). For each invocation the entire input residual file is scanned until the selected iteration header line is found (e.g., "Iteration 0" or "Final Iteration"). Then each GCP record for that iteration is loaded into a residual data structure containing the fields:

1. Point Id
2. Predicted L1G Line
3. Predicted L1G Sample
4. GCP Observation Time (seconds)
5. GCP Latitude (degrees)
6. GCP Longitude (degrees)
7. GCP Height (meters)
8. Across-Track View Angle (degrees)
9. Across-Track Residual (meters)
10. Along-Track Residual (meters)
11. Image Y Residual (meters)
12. Image X Residual (meters)
13. Outlier/Valid Flag (0 = outlier, 1 = valid)
14. GCP Source (DOQ or GLS)

Note that the inclusion of GCP source in the precision residual file and in the input residual structure is new for LDCM.

Remove Outliers Sub-Algorithm (remove_outliers)

This function removes the residual records flagged as outliers from the GCP residual information and places the data in buffers to be accessed by later routines. Records with the outlier/valid flag field set to 0 are outliers.

Analyze GCP Residuals Sub-Algorithm (analyze_GCP_res)

This function calculates the mean, root mean square error (RMSE), and standard deviation of the along- and across-track GCP residuals as well as the correlation coefficient between the across- and along-track residuals. The statistics are computed in the following process:

a) Calculate GCP statistics
    a1) Calculate total number of GCPs used (count of valid GCPs)
    a2) Calculate mean latitude of GCPs used
    a3) Calculate mean longitude of GCPs used

b) Calculate pre-fit statistics
    b1) Calculate mean of cross-track residuals
    b2) Calculate mean of along-track residuals
    b3) Calculate RMSE of cross-track residuals
    b4) Calculate RMSE of along-track residuals
    b5) Calculate standard deviation of cross-track residuals

b6) Calculate standard deviation of along-track residuals
b7) Calculate correlation coefficient between along- and cross-track residuals

c) Calculate post-fit statistics
c1) Calculate mean of cross-track residuals
c2) Calculate mean of along-track residuals
c3) Calculate RMSE of cross-track residuals
c4) Calculate RMSE of along-track residuals
c5) Calculate standard deviation of cross-track residuals
c6) Calculate standard deviation of along-track residuals
c7) Calculate correlation coefficient between along- and cross-track residuals

The following equations are used to perform these calculations, with X being the parameter for which statistics are calculated:

Mean (ALIAS xxx_mean):

$$X_{mean} = \frac{1}{numGCP} \sum_{i=1}^{numGCP} X_i$$

RMSE (ALIAS xxx_RMSE):

$$X_{RMS} = \sqrt{\frac{1}{numGCP} \sum_{i=1}^{numGCP} X_i^{\,2}}$$

Standard Deviation (ALIAS xxx_std_dev):

$$X_{StdDev} = \sqrt{\frac{1}{numGCP-1}\left(\left(\sum_{i=1}^{numGCP} X_i^{\,2}\right) - numGCP * X_{mean}^{\,2}\right)}$$

Correlation Coefficient (ALIAS xxx_corr_coeff):

$$XY_{CC} = \frac{\left(\sum_{i=1}^{numGCP}\left(X_i - X_{mean}\right)\left(Y_i - Y_{mean}\right)\right)}{\left(numGCP-1\right)} \frac{1}{X_{StdDev} Y_{StdDev}}$$

Output Residual Statistics Sub-Algorithm (output_resid_stats)
This function creates the output geodetic report file and writes the statistics computed from the GCP residuals to the output file. Note that the output of trending data to the characterization database is performed by the geodetic characterization main procedure.

Write Residual Statistics Sub-Algorithm (write_resid_stats)
This function invokes output_header to write the standard report file header and then writes the GCP residual statistics to the ASCII output file.

Write Standard Report Header Sub-Algorithm (output_header)
This function collects the input parameters, image metadata, and environment variable values needed to populate the standard IAS report header and writes the header to the ASCII output file.

Algorithm Output Details

The geodetic accuracy assessment algorithm outputs are summarized in Table 1 below. All fields are written to the output report file. Only those with "Yes" in the "Database Output" column are written to the characterization database. Note that the first eleven fields listed constitute the standard report header. Also note that the DEM Source field present in the heritage ALIAS implementation is no longer required.

Field	Description	Database Output
Date and time	Date (day of week, month, day of month, year) and time of file creation.	Yes
Spacecraft and instrument source	LDCM and OLI	Yes
Processing Center	Processing center where the output was generated (see note #4).	Yes
Work order ID	Work order ID associated with processing (blank if not applicable).	Yes
WRS path	WRS path number	Yes
WRS row	WRS row number	Yes
Software version	Software version used to create report	Yes
Roll angle	Scene off-nadir roll angle (in degrees)	Yes
Acquisition Type	Earth viewing, Lunar, or Stellar (only Earth-viewing scenes are used for geodetic characterization)	Yes
L0Rp ID	Input L0Rp image ID	Yes
L1G image file	Name of L1G used to measure GCPs	No
Acquisition date	Date of L1G image acquisition	Yes
GCP source	Source of GCPs (GLS or DOQ)	Yes
Number of valid points	Number of GCPs accepted as valid	Yes
Mean GCP latitude	Mean latitude of valid GCPs (degrees)	Yes
Mean GCP longitude	Mean longitude of valid GCPs (degrees)	Yes
Pre-fit along-track mean	Mean of along-track iteration 0 residuals (meters)	Yes
Pre-fit across-track mean	Mean of across-track iteration 0 residuals (meters)	Yes
Pre-fit along-track RMSE	RMSE of along-track iteration 0 residuals (meters)	Yes
Pre-fit across-track RMSE	RMSE of across-track iteration 0 residuals (meters)	Yes
Pre-fit along-track standard deviation	Standard deviation of along-track iteration 0 residuals (meters)	Yes

Pre-fit across-track standard deviation	Standard deviation of across-track iteration 0 residuals (meters)	Yes
Pre-fit correlation coefficient	Correlation coefficient between along- and across-track iteration 0 residuals (dimensionless)	Yes
Post-fit along-track mean	Mean of along-track final iteration residuals (meters)	Yes
Post-fit across-track mean	Mean of across-track final iteration residuals (meters)	Yes
Post-fit along-track RMSE	RMSE of along-track final iteration residuals (meters)	Yes
Post-fit across-track RMSE	RMSE of across-track final iteration residuals (meters)	Yes
Post-fit along-track standard deviation	Standard deviation of along-track final iteration residuals (meters)	Yes
Post-fit across-track standard deviation	Standard deviation of across-track final iteration residuals (meters)	Yes
Post-fit correlation coefficient	Correlation coefficient between along- and across-track final iteration residuals (dimensionless)	Yes

Table 1: Geodetic Accuracy Assessment Output Details

Accessing the Geodetic Accuracy Characterization Database

Though not part of the formal geodetic accuracy assessment algorithm, some comments regarding the anticipated methods of accessing and analyzing the geodetic accuracy results stored in the characterization database may assist with the design of the characterization database.

The database output from the geodetic accuracy assessment algorithm will be accessed by a statistical summary analysis tool that queries the characterization database to retrieve geodetic accuracy results from multiple scenes. Summary mean and RMSE statistics for the pre-fit scene results will be calculated and output in a report containing a comma-delimited table of the retrieved trending results as well as the summary statistics.

The geodetic results would typically be queried by acquisition date, roll angle, WRS path/row, and/or GCP source. The most common query would be a combination of GCP source, roll angle, and acquisition date range, for example, selecting all of the GLS-derived results, from nadir scenes, for a given calendar quarter:

GCP_Source = "GLS"
Roll_Angle is between -0.5 and 0.5
Acquisition_Date is between 01APR2012 and 30JUN2012

Since we will be using the roll angle field to detect off-nadir acquisitions, it would be convenient if the associated query could be specified as a maximum (absolute) number (e.g., 0.5 degrees) rather than having to specify a plus/minus range.

The summary mean and RMSE statistics would be calculated from the pre-fit and post-fitmean and RMSE results for the individual scenes returned as:

$$Mean_{net} = \frac{1}{numScene} \sum_{i=1}^{numScene} Mean_i$$

$$RMSE_{net} = \sqrt{\sum_{i=1}^{numScene} RMSE_i^{\,2} \Big/ numScene}$$

The query results would be formatted in a set of comma-delimited records (for ease of ingest into Microsoft Excel), one record per scene. Each record would contain all of the fields written to the characterization database (items with "Yes" in the rightmost column of Table 1 above). A header row containing the field names should precede the database records. Two trailer rows, one containing the summary statistic names (Net Pre-fit Along-Track Mean, Net Pre-fit Across-Track Mean, Net Pre-Fit Along-Track RMSE, Net Pre-Fit Across-Track RMSE, Net Post-Fit Along-Track Mean, Net Post-Fit Across-Track Mean, Net Post-Fit Along-Track RMSE, and Net Post-Fit Across-Track RMSE) and the second containing the comma-delimited summary statistic values, should follow the database records.

7.2.7.8 Maturity

1. The computation of GCP statistics is essentially the same as what is currently used in the ALIAS heritage code, and was used in the Landsat 7 IAS implementation.
2. A field to capture the GCP source (GLS vs. DOQ) has been added to the GCP residual record.

7.2.7.9 Notes

Some additional background assumptions and notes include:

7. The pre-fit mean and RMSE statistics derived from DOQ control capture the absolute geodetic accuracy performance of the LDCM system whereas the standard deviation statistics reflect the relative geodetic accuracy.
8. The post-fit RMSE statistics provide an indication of the absolute accuracy of the output L1T product but this must be combined with an assessment of the accuracy of the GCP source to obtain a more realistic estimate of L1T accuracy. L1T accuracy is measured directly by the geometric accuracy assessment algorithm, which is a variant of the geodetic accuracy assessment process but is documented as a separate algorithm (see Geometric Accuracy Assessment ADD).
9. The per-scene post-fit along-track RMSE and post-fit across-track RMSE statistics, derived from the GLS control used for L1T product generation, would be good candidates for use as geometric quality metrics. The post-fit RMSE statistics could be extracted from either the geodetic accuracy assessment report file or the characterization database. In the case of a LOS model correction failure, fill values would be inserted into these quality fields to indicate that the registration to the GLS control failed, for example, due to cloud cover.
10. A configuration table (system table) and/or environment variables should be provided for each installation of the algorithm implementation to convey site-specific information such as the processing center name (used in the standard report header), the number of processors available (for parallel processing implementations), etc. This takes the place of the heritage system table which also contained certain algorithm-related parameters. Anything related to the algorithms has been moved to the CPF for LDCM.

11. The input L1G image is only used to extract selected metadata (noted in the input table) for inclusion in the output report and trending data. If the required fields are all available in the L0Rp data set, it could be used as an input instead of the L1G. Since, unlike the L1G, the geometry team has no control over the contents of the L0Rp, we leave this as a design trade for the operational software.

7.2.8 OLI Image Registration Accuracy Assessment Algorithm

7.2.8.1 Background/Introduction

The OLI Image Registration Accuracy Assessment, or image-to-image (I2I) characterization, algorithm has two purposes; it can be used to determine the geometric registration of an image to a particular source image or it can be used to verify the multi-temporal capabilities of the OLI product generation system.

The I2I characterization process works by choosing locations within the reference and search images, extracting windows of imagery from each image and performing grey scale correlation on the image windows. Several criteria are used in determining whether the correlation processing was successful. These criteria include measured displacement and strength of the correlation peak. The subpixel location of the measured offset is calculated by fitting a 2^{nd} order polynomial around the correlation surface and solving for the fractional peak location of the fitted polynomial. Once the total offset has been measured, adding the calculated integer offset to the calculated subpixel offset, for all successfully correlated tie-points a final t-distribution outlier rejection is performed to produce the set of all valid measured offsets.

There are two options available for determining tie-point locations in the I2I characterization algorithm. These options include choosing evenly spaced points in the output space of the imagery or choosing points based on pre-chosen locations.

The OLI Image Accuracy Assessment algorithm is derived from the corresponding Advanced Land Imagery (ALI) I2I characterization algorithm used in ALI Image Assessment System (ALIAS). Its implementation should be very similar to the ALI I2I characterization application. The algorithm will have to be modified to accommodate LDCM data formats.

7.2.8.2 Dependencies

The OLI I2I algorithm assumes a cloud free L1T has been generated and that a suitable reference image (OLI or other source) exists for comparison purposes. The L1T needs to be in the SCA combined format.

7.2.8.3 Inputs

The Image Accuracy Assessment algorithm and its component sub-algorithms use the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs	**Source**
Reference image	ODL
Search image	ODL
Bands to process	ODL
Tie point type	ODL
Number points in line direction (if tie-point type is evenly spaced)	ODL
Number points in sample direction (if tie-point type is evenly spaced)	ODL
GCP File Name (if tie-point type is file based)	ODL
Correlation window size lines	CPF/ODL
Correlation window size samples	CPF/ODL

Fill range maximum	CPF/ODL
Fill range minimum	CPF/ODL
Fill percentage	CPF/ODL
T-distribution outlier threshold	ODL
Output file names	
I2I residuals file (see Table #2 below for details)	ODL
I2I output data file (see Table #1 below for details)	ODL
I2I statistics file (see Table #3 below for details)	ODL
Trend flag	ODL
L0R/L1R ID	
Work Order ID	
WRS Path/Row	
Trending thresholds (RMS for each line,sample per band – See note #5)	CPF
Minimum correlation peak	CPF/ODL
Maximum displacement	CPF/ODL
Correlation fit method (See note #2)	CPF

7.2.8.4 Outputs

I2I residuals file (See table #2)
I2I data file (See table #1)
I2I statistics file (See table #3)
I2I characterization trending (if trending flag set to yes)
L0R/L1R ID
Work Order ID
WRS Path/Row
Reference source
I2I statistics (Min, Mean, Max, Median, RMS, Std. Dev.)

The following processing parameters that are listed in the table above can be overridden if they are given as fields within the input ODL file; correlation window size, maximum offset, minimum correlation strength, fill threshold, maximum and minimum file values.

7.2.8.5 Options

Trending on/off switch
Correlation fit method (placeholder, see note #2)
Normalized grey scale or least squares correlation (see note #2)

7.2.8.6 Prototype Code

Input to the executable is an ODL file; output is an ASCII file containing measured offsets between the input image file and GCP library. The prototype output/input directory contains the input ODL files needed, the HDF5 input image files, the GCPLib look-up file, and the CPF.

The prototype code was compiled with the following options when creating the test data files:

```
-g -Wall -march=nocona -m32
```

The following are brief descriptions of the main set of modules used within the prototype. It should be noted that almost all library modules are not referenced in the explanations below. Only those

modules within the main i2ichar directory for the prototype and any library modules that were determined to be important to the explanation of the results, input parameters, or output parameters, are discussed.

i2i_char
Main driver for the image registration assessment. Calls modules to read input parameters (get_parms), check image characteristics (compare_metadata), create tie-point locations (create_tiepts and read_gcps), and perform image registration mensuration (i2i_corr).

get_parms
Reads the parameters from the input ODL parameter file and retrieves the I2I default processing parameters from the CPF; correlation fit method, minimum correlation peak, maximum allowable displacement.

i2i_corr
Main driver for image correlation, or image mensuration, process. Driver opens image, calls module to perform correlation at tie-point locations (process_gcp), and writes out image registration residuals file (table 1).

process_gcp
Process to perform correlation between one band of the search and reference images. Initializes and calls correlation libraries, extracts image data from files, calls module ias_math_check_pixels_range to determine if a given window of imagery contains enough "valid-range" pixels so that mensuration can be performed.

ias_math_check_pixels_range
Checks to see if percent of pixels within a given buffer is within a set range of values. Range and percentage is a user defined parameter.

compare_metadata
Checks image file metadata for validity of data with regards to image registration assessment. Checks include; bands requested for assessment being present, same pixel size between images, same map projection, projection parameters, projection spheroid, and datum. Messaging is also present to warn users when the two images to be assessed do not have the appropriate correction type. Image and band metadata structures are passed back so so that it can be used by other modules.

create_tiepts
Main driver for generating tie-points based on evenly space tie-point option. Calls module i2i_det_tiepts.

i2i_det_tiepts
Calculates tie-point locations for evenly spaced tie-point option. Calls module math_calc_poly (twice) to determine mapping polynomials between map coordinates and line/sample locations for search and reference images. Calls module xxx_eval to map corner coordinates to line and sample locations.

math_calc_poly

Uses image metadata information to determine mapping polynomial between map coordinates and line/sample locations for an image file.

read_gcps
Opens GCP ASCII, or within the prototype code the GCPLib, file. Calls module io_get_gcplib to read ASCII GCP (GCPLib) information. Calls module misc_gcp_ls to map ASCII GCP (GCPLib) geographic locations to search and reference line and sample locations.

io_get_gcplib
Reads GCPLib file storing GCP information according to requested criteria; absolute, relative, begin and end date, season, chip source.

misc_gcp_ls
Converts geographic location of GCPs to line and sample location within search image.

read_metadata
Reads file and band metadata for imagery.

7.2.8.7 Procedure

Image-to-image (I2I) characterization is used to assess the ability to register an OLI data set to another image data set. I2I characterization performs image correlation between OLI imagery, the search data set, and a reference image data set. Landsat, reduced resolution DOQ data, or OLI imagery can be used as a reference data set. When OLI imagery is measured against another OLI data set acquired at a different date over the same geographic area, I2I measures the ability to register multi-temporal OLI imagery. Correlation points are chosen either as evenly distributed points throughout the imagery or at predefined GCP locations. Outliers are first rejected by removing all measured displacements that lie above a user set threshold or those whose correlation peak is below a given minimum value. A final outlier rejection is performed on the measured offsets using a Student-t distribution test. Final statistics, which are reported in the output statistics file and stored in the database, are calculated based on the valid displacements after the Student-t outlier rejection. Statistics are calculated for both the line and sample direction independently. An overview of the algorithm procedure is shown in figure 1.

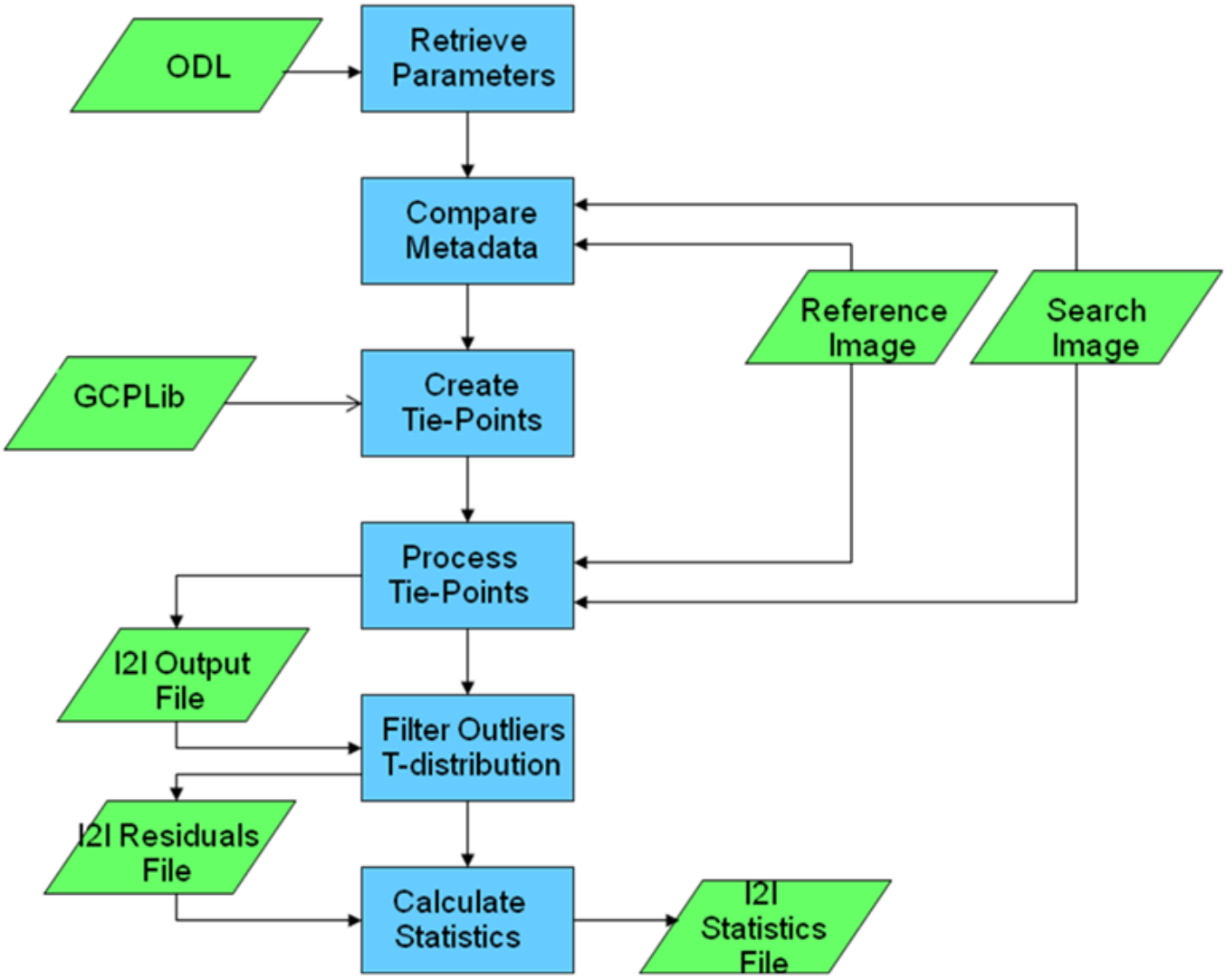


Figure 1: Image Registration Accuracy Assessment Algorithm Flow Diagram

The OLI Image Registration Accuracy Assessment algorithm has heritage in the Landsat 7 (L7) and Advanced Land Imagery (ALI) Image Assessment Systems (IAS) Image-to-Image Characterization (I2I Char) algorithm/process. The prototype code for OLI Image Registration Accuracy Assessment will contain many of the same modules that are present in the L7 and ALI IAS I2I Characterization. The correlation and mensuration modules however are not described within this ADD as they are already presented in the Ground Control Correlation and Band Registration Accuracy Assessment ADDs. Those ADDs should be reviewed for any information pertaining to these processes. Explanations of the methodology of the mensuration and t-distribution outlier rejection processes are presented in the Band Registration Accuracy Assessment ADD. That ADD should be reviewed for any information pertaining to these methodologies.

7.2.8.7.1 Stage 1 - Data Input

The data input stage involves loading the information required to perform the image registration assessment. This includes reading the image files, retrieving the output I2I file names: data, residuals and statistic files; retrieving or initializing processing parameters: maximum displacement, fill range, fill threshold, minimum correlation peak, t-distribution threshold, bands to process,

correlation window size, trendingthresholds, tie-point method; and if tie-point method is set to file-based the GCP file name. Once the input file, and if need be the GCP name, has been retrieved the files and the information stored within them can be opened and read.

7.2.8.7.2 Stage 2 - Create Tie-point Locations

Tie point locations may be determined in an evenly spaced pattern in output space or they may be read from a GCP file.

Determine Evenly Spaced Tie-points

This tie-point selection process is similar to the Band Registration Accuracy Assessment ADD, Stage 3 - Create Tie-point Locations section Determine Evenly Spaced Tie-points. The difference between the two processes is that the Image Accuracy Assessment algorithm computes and uses the bounding area between the search and references for tie-point selection.

Creating Evenly Space Tie-Points Processing Steps

1. Map search corners to reference space

$$\text{search line}_i = \frac{Y_{ULref} - Y_i}{P_y} + 1$$

$$\text{search sample}_i = \frac{X_i - X_{ULref}}{P_x} + 1$$

Where
i = 0,1,2,3 for the search upper left, upper right, lower right, lower left coordinates
Y_{ULref} = Reference upper left Y coordinate
X_{ULref} = Reference upper left X coordinate
P_x = pixel size sample direction
P_y = pixel size line direction

2. Determine bounding overlapping area.
minimum sample = min(search sample_i, reference sample_i)
maximum sample = max(search sample_i, reference sample_i)
minimum line = min(search line_i, reference line_i)
maximum line = max(search line_i, reference line_i)

3. Calculate step sizes

$$\text{spacing x} = \frac{\text{maximum sample} - \text{minimum sample} - \text{correlation window samples}}{\text{M} - 1}$$

$$\text{spacing y} = \frac{\text{maximum line} - \text{minimum line} - \text{correlation window lines}}{\text{N} - 1}$$

Where
M = Number of tie-points in sample direction
N = Number of tie-points in line direction

4. Set evenly spaced tie-point locations
   4.1 For j = 0 to N-2

$$\text{tie-point location y}[\text{j}] = \frac{\text{correlation window lines}}{2} + \text{j} * \text{spacing y}$$

4.2 $\text{tie-point location } y[N-1] = \text{maximum line} - \frac{\text{correlation window lines}}{2}$

4.3 For i = 0 to M-2

$$\text{tie-point location } x[i] = \frac{\text{correlation window samples}}{2} + i * \text{spacing } x$$

4.4 $\text{tie-point location } x[M-1] = \text{maximum sample} - \frac{\text{correlation window samples}}{2}$

Determine File-Based Tie-points

This tie-point selection is based on an input ASCII file containing latitude and longitude locations for each individual tie-point. These individual locations are converted to line and sample locations within the search and reference images. These locations then have windows of imagery extracted from the search and reference images after which displacement between the two windowed images can be calculated. This file called the GCPLib within prototype and referred to as the ASCII GCP file within the text of this document.

Create Tie-Point from GCP ASCII (GCPLib) file Processing Steps.

1. Open GCP ASCII (GCPLib) file.

2. For each GCP

2.1 read GCP (note #1).
Chip ID and name
Latitude and longitude
Projection X, Y, Z

2.2 Convert GCP geographic/projection location to line and sample locations within image files.

2.2.1 Convert GCP geographic location to search/reference map projection. Map projection conversions can be done through General Cartographic Transformation Package (GCTP).

2.2.2 Convert map projection X and Y locations to line and sample locations.

Equations:

$$line = \frac{Y_{GCP} - Y_{UL}}{P_y} + 1$$

$$sample = \frac{X_{UL} - X_{GCP}}{P_x} + 1$$

Where
Y_{UL} = Upper left Y coordinate of image
X_{UL} = Upper left X coordinate of image
Y_{GCP} = Y coordinate of GCP
X_{GCP} = X coordinate of GCP
P_x = Image pixel size in sample direction

P_y = Image pixel in size in line direction

2.2.2.a Convert to line and sample location in search image.
2.2.2.b Convert to line and sample location in reference image.

2.3 Store line and sample locations for search and reference.

3 Close GCP ASCII (GCPLib) file

7.2.8.7.3 Stage 3. Calculate Individual Point-by-Point Image Displacements

Normalized cross correlation is used to measure spatial differences between the reference and search windows extracted from the imagery to be compared. The normalized cross correlation process helps to reduce any correlation artifacts that may arise from radiometric differences between the two image sources. The correlation process will only measure linear distortions over the windowed areas. By choosing appropriate correlation windows that are well distributed throughout the imagery, nonlinear differences between the image sources can be found. This methodology is explained in the Band Registration Accuracy Assessment ADD. The processes and modules associated with these calculations are explained in the GCP Correlation and Band Registration Accuracy Assessment ADDs.

7.2.8.7.4 Stage 4. Removing Outliers Using the t-distribution

Once all the line and sample offsets have been measured and the first level of outlier rejection has been performed, a check against the maximum allowable offset and the minimum allowable correlation peak, the measurements are further filtered for outliers using a Student-t outlier rejection. This methodology, along with the processes and modules that are present within ALIAS, are explained in the Band Registration Accuracy Assessment ADD. That ADD should be used as a reference for these items.

Image Accuracy Assessment Processing Steps.

1. For band = Number of OLI bands to process

    1.1. For index = Number of tie-points to process

        1.1.1. Read current tie-point chip and tie-point location
        Set tie-point flag to unsuccessful

        1.1.2. Extract search window (of imagery) at tie-point location

        1.1.3. Extract reference window (of imagery) at tie-point location

        1.1.4. Count number of pixels in reference window that is within fill range.
        count = 0
        For i=0 to number of pixels in correlation window
            If reference pixel is > fill min and reference pixel is < fill max
                count++

1.1.5. Check number of reference pixels counted against fill threshold/percentage.

$$\text{if } \frac{\text{count}}{\text{correlation window size}} > \text{fill threshold}$$

increment index to next tie-point location
else
continue

1.1.6. Count number of pixels in search window that is within fill range.
count = 0
For i=0 to number of pixels in correlation window
If search pixel is > fill min and search pixel is < fill max
count++

1.1.7. Check number of search pixels counted against fill threshold/precentage.

$$\text{if } \frac{\text{count}}{\text{correlation window size}} > \text{fill threshold}$$

increment index to next tie-point location
else
continue

1.1.8. Perform normalized grey scaled correlation between reference and search windowed images, calculating correlation surface *R* (See Band Registration Accuracy Assessment ADD-Stage 4 Calculate Individual Point-by-Point Band Displacements).

1.1.9. Find peak within correlation surface
Max = *R*(0,0)
For i=0 to correlation window number of lines -1
For j=0 to correlation window number of samples -1
If *R*(i,j) > max then
Max = *R*(i,j)
line offset = i
sample offset = j

1.1.10. Check correlation peak against threshold
if max > peak threshold
continue
else
set tie-point flag to outlier and choose next tie-point

1.1.11 Measure sub-pixel peak location (See Band Registration Accuracy Assessment ADD - Stage 4 Calculate Individual Point-by-Point Band Displacements)
Δsub-line
Δsub-sample

1.1.12. Calculate total pixel offset
total line offset = line offset + Δsub-line
total sample offset = sample offset + Δsub-sample

1.1.13. Check offset against maximum displacement offset

$$\text{total displacement} = \sqrt{(\text{total line offset})^2 + (\text{total sample offset})^2}$$

if (total displacement > maximum displacement)
Set tie-point flag to outlier and choose next tie-point
else
Set tie-point flag to valid

1.2 Store band tie-point mensuration information, correlation success, and offsets measured. See table #1.

3. For band = 1 to Number of bands to process

3.1 Perform t-distribution outlier rejection (See Band Registration Accuracy Assessment - Stage 5 Removing Outliers Using the t-distribution).

3.2. Store band combination final individual tie-point information and outlier flag. See table #2.

4. For band combination = 1 to Number of band combinations

4.1. Calculate mean, minimum, maximum, median, standard deviation, and root mean squared offset.

4.2. Store band combination statistics. See table #3.

5. Perform trending if trending flag is set to yes

5.1 Check results against trending thresholds
For each band
if measured RMSE > trending thresholds
exit trending
If there are no RMSE > trending thresholdsperform trending

7.2.8.7.5 Output files

The output files listed below for the Image Registration Accuracy Assessment follow the philosophy of the Advanced Land Imager Assessment System (ALIAS) Image-to-Image (I2I) Characterization output files in that they are made generic so that the same format can be used elsewhere.

All output files contain a standard header. This standard header is at the beginning of the file and contains the following:

1) Date and time file was created.
2) Spacecraft and instrument pertaining to measurements.
3) Off nadir (roll) angle of spacecraft/instrument.
4) Acquisition type
5) Report type (Image-to-Image)
6) Work order ID of process (left blank if not applicable)
7) WRS path/row

8) Software version that produced report.
9) L0R image file name

The data shown within Table 3 listed below is stored in the database. The statistics stored per band will be used for trending analysis of the image registration accuracy of the OLI instrument. Results produced through a time-series analysis of this data stored, over a set time interval or multiple image files, will be used for a temporal assessment of the registration quality of the OLI products. The SCA number fields are listed in the tables for Image Accuracy Assessment for consistency with the tables listed in the Band Accuracy Assessment ADD.

99. Field	**100. Description**
101. Date and time	102. Date (day of week, month, day of month, year) and time of file creation.
103. Spacecraft and instrument source	104. LDCM and OLI (TIRS if applicable)
105. Processing Center	106. EROS Data Center SVT
107. Work order ID	108. Work order ID associated with processing (blank if not applicable)
109. WRS path/row	110. WRS path and row
111. Software version	112. Software version used to create report
113. Off-nadir angle	114. Off-nadir roll angle of processed image file
115. Acquisition Type	116. Earth viewing or Lunar
117. L0R image file	118. L0R image file name used to create L1T
119. Processed image file name	120. Name of L1T used to create report
121. Reference bands	122. Reference bands used in image assessment
123. Search bands	124. Search bands used in image assessment
125. Heading for individual tie-points	126. One line of ASCII text defining individual tie-point fields.
127. For each tie-point:	128.
129. Tie point number	130. Tie-point index/number in total tie-point list
131. Reference line	132. Tie-point line location in reference image (band)
133. Reference sample	134. Tie-point sample location in reference image (band)
135. Reference latitude	136. Tie-point latitude location
137. Reference longitude	138. Tie-point longitude location
139. Reference elevation	140. Elevation of tie-point location
141. Search line	142. Tie-point line location in search image
143. Search sample	144. Tie-point sample location in search image
145. Delta line	146. Measured offset in line direction
147. Delta sample	148. Measured offset in sample direction

149. Outlier flag	150. 1=Valid, 0=Outlier
151. Reference band	152. Reference band number
153. Search band	154. Search band number
155. Reference SCA	156. SCA number that reference window was extracted from
157. Search SCA	158. SCA number that search window was extracted from
159. Search image	160. Name of search image
161. Reference image	162. Name of reference image

Table 3. Image Registration Accuracy Assessment Data File

163. Field	**164. Description**
165. Date and time	166. Date (day of week, month, day of month, year) and time of file creation.
167. Spacecraft and instrument source	168. LDCM and OLI (TIRS if applicable)
169. Processing Center	170. EROS Data Center SVT
171. Work order ID	172. Work order ID associated with processing (blank if not applicable)
173. WRS path/row	174. WRS path and row
175. Software version	176. Software version used to create report
177. Off-nadir angle	178. Off-nadir pointing angle of processed image file
179. Acquisition Type	180. Earth viewing or Lunar
181. L0R image file	182. L0R image file name used to create L1T
183. Processed image file name	184. Name of L1T used to create report
185. Number of records	186. Total number of tie-points stored in file
187. Heading for individual tie-points	188. One line of ASCII text defining individual tie-point fields.
189. For each band combination	190.
191. Combination header	192. Number of points in combination, reference band number, search band number.
193. For each tie-point:	194.
195. Tie point number	196. Tie-point index/number in total tie-point list

197. Reference line	198. Tie-point line location in reference image (band)
199. Reference sample	200. Tie-point sample location in reference image (band)
201. Reference latitude	202. Tie-point latitude location
203. Reference longitude	204. Tie-point longitude location
205. Reference elevation	206. Elevation of tie-point location
207. Search line	208. Tie-point line location in search image
209. Search sample	210. Tie-point sample location in search image
211. Delta line	212. Measured offset in line direction
213. Delta sample	214. Measured offset in sample direction
215. Outlier flag	216. 1=Valid, 0=Outlier
217. Reference band	218. Reference band number
219. Search band	220. Search band number
221. Reference SCA	222. SCA number that reference window was extracted from
223. Search SCA	224. SCA number that search window was extracted from
225. Search image	226. Name of search image
227. Reference image	228. Name of reference image

Table 4. Image Registration Accuracy Assessment Residuals File

229. Field	**230. Description**
231. Date and time	232. Date (day of week, month, day of month, year) and time of file creation.
233. Spacecraft and instrument source	234. LDCM and OLI (TIRS if applicable)
235. Processing Center	236. EROS Data Center SVT
237. Work order ID	238. Work order ID associated with processing (blank if not applicable)
239. WRS path/row	240. WRS path and row
241. Software version	242. Software version used to create report
243. Off-nadir angle	244. Off-nadir pointing angle of processed image file
245. Acquisition	246. Earth viewing or Lunar

Type	
247. L0R image file	248. L0R image file name used to create L1T
249. Processed image file name	250. Name of L1T used to create report
251. t-distribution threshold	252. Threshold used in t-distribution outlier rejection
253. For each band	254.
255. Reference band	256. Reference band of statistics
257. Search band	258. Search band of statistics
259. SCA	260. SCA number of search image
261. Total tie-points	262. Total number of tie-points for band
263. Correlated tie-points	264. Number of tie-points that successfully correlated for band
265. Valid tie-points	266. Total number of valid tie-points for band after all outlier rejection has been performed
267. For both line and sample direction:	268. All statistics are given in terms of pixels
269. Minimum offset	270. Minimum offset within all valid offsets
271. Mean offset	272. Mean offset of all valid offsets
273. Maximum offset	274. Maximum offset within all valid offsets
275. Median offset	276. Median offset within all valid offsets
277. Standard deviation	278. Standard deviation of all valid offsets
279. Root-mean-squared	280. Root mean squared offset of all valid offsets

Table 5. Image Registration Accuracy Assessment Statistics Output File

7.2.8.8 Maturity

3. Currently the t-distribution outlier rejection happens as a completely separate process. This is due only to Landsat heritage where outlier rejection was done through the Analyst User Interface (AUI).

7.2.8.9 Notes

Some additional background assumptions and notes include:

12. The GCP structure and retrieval modules are setup to be generic. This structure contains the following for each GCP:

1. a) Point ID
2. b) Chip name
3. c) Reference line and sample

4. d) Latitude, Longitude
5. e) Projection X, Y, Z
6. f) Pixel size Y, X
7. g) Image chip size line, sample
8. h) Source of GCP
9. i) Date of GCP
10. j) Relative/absolute flag
11. This type of generic GCP structure and management will work for OLI processing also.
13. The correlation result fit method defines the algorithm used to estimate the correlation peak location to sub-pixel accuracy. Only the quadratic surface fitting method described in this ADD is supported in the baseline algorithm. Note that the fine least-squares correlation method, invoked by selecting correlation windows with an odd number of lines or samples, does not use a separate peak finding method.
14. Image Registration Accuracy statistics stored within the database will be accessed for analysis.
    a. *Accessed according to a specific date range.*
    b. *Accessed according to a specific band.*
    c. *Accessed according to a specific geographic location.*
    d. *Accessed according to acquisition type (nadir, off-nadir, lunar).*

This data accessed can be retrieved and stored within a comma delimited file. The methodology used to access the database could be an SQL script.

15. Data stored within the database will be accessed for time series analysis.
    a. *Data would be pulled for a user-specified time period.*
    b. *Statistics over multiple scenes would be calculated and combined into band or scene based statistics.*

These calculations could be performed within the methodology used to access the data from the database (SQL script).

16. There will need to be a set of criteria, based on calculated statistics, as to whether trending should be performed or not. These criteria would be provided to avoid having garbage stored in the database. Any values needed in determining whether the criteria have been met for trending would be stored and retrieved from the CPF. There would be one threshold per band. The criteria to check for trending are shown in section 5.1 of the Image Accuracy Assessment Processing steps section.
17.

7.2.9 OLI Band Registration Accuracy Algorithm

7.2.9.1 Background/Introduction

The OLI Band Registration Accuracy Assessment Algorithm (BRAA), or the Band-to-Band (B2B) Characterization process, measures the relative band alignment between all bands of each Sensor Chip Assembly (SCA) for the OLI instrument. The displacement for every pair-wise combination of all bands of each SCA requested for assessment is measured; creating an over determined data set of band-to-band measurements for each SCA. The residuals measured from the B2B characterization process will be used to assess the accuracy of the band-to-band registration of the OLI instrument, and if need be, used as input to the band calibration algorithm in order to calculate new line-of-sight (LOS) parameters for the Calibration Parameter File (CPF).

The B2B characterization process works by choosing tie point locations within band pairs of each SCA, extracting windows of imagery from each band and performing grey scale correlation on the image windows. Several criteria are used in determining whether the correlation processing was

successful. These criteria include measured displacement and strength of the correlation peak. The sub-pixel location of the measured offset is calculated by fitting a 2nd order polynomial around the discrete correlation surface and solving for the fractional peak location of the fitted polynomial. The total offset measured is then the integer location of the correlation peak plus the sub-pixel location calculated. A new fine-resolution least squares correlation method has been added to the heritage algorithm to provide more accurate measurement of sub-pixel offsets. This method is described below.

There are several options available for processing data through the Band Registration Accuracy Assessment algorithm. These include choosing evenly spaced points for location of the windows extracted, choosing to use the geometric grid for determining window locations in order to avoid fill within the image files, specifying the bands and/or the SCA to process, and specifying the valid pixel range to use during correlation. The least squares correlation method is invoked by requesting image windows with at least one odd dimension , since the heritage algorithm only works with images with even dimensions (e.g., 32x32 image windows will use normalized grey scale correlation but 31x31 image windows will use least squares correlation).

An Earth based acquisition will be used to characterize all bands except the cirrus. A lunar acquisition will be used to characterize the cirrus band. Both types of acquisitions will be passed through BRAA. In terms of the BRAA it does not matter which type of acquisition is being passed into the algorithm, some of the processing parameters and options may change due to the acquisition type but both types will use the same mensuration process to create an assessment of the band registration.

7.2.9.2 Dependencies

The OLI BRAA assumes that a cloud free Earth viewing L1T or Lunar L1G image has been generated and depending on the tie point selection type chosen, that the LOS Model Correction and the LOS Projection and Gridding algorithms have been executed to create a geometric grid file. The L1T/L1G image needs to be in the SCA-separated format and either in a SOM or UTM path-oriented projection for Earth acquisitions. The digital orthophoto quadrangle (DOQ) control and best available digital elevation model (DEM) needs to be used in generating the L1T.

7.2.9.3 Inputs

The BRAA and its component sub-algorithms use the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs	**Source**
ODL file (implementation)	
Calibration Parameter File (baseline)	ODL
Correlation Fit Method (see note #14)	CPF
Correlation Window Size	CPF or ODL
Correlation Maximum Displacement	CPF or ODL
Correlation Fill Threshold	CPF or ODL
Correlation Minimum Fill Value	CPF or ODL
Correlation Maximum Fill Value	CPF or ODL
L1T/L1G image	ODL
OLI resampling grid (optional)	ODL
Outlier (t-distribution) threshold	ODL
B2B characterization output file	ODL

Output residuals file name	ODL
Output statistics file name	ODL
SCAs to process	ODL
Bands to process	ODL
Fill range maximum	ODL
Fill range minimum	ODL
Fill threshold or percentage	ODL
Correlation window size lines	ODL
Correlation window size samples	ODL
Tie-point spacing in line direction	ODL
Tie-point spacing in sample direction	ODL
Trending flag	ODL
L0R ID (for trending)	ODL
Work Order ID (for trending)	ODL
Calibration Parameter File (baseline, if trending is requested)	ODL
Trending thresholds (Standard deviation line, sample per band per SCA - see note #3).	CPF

7.2.9.4 Outputs

Pan downsampled image
B2B residuals file (see Table 2 below for details)
B2B output data file (see Table 1 below for details)
B2B statistics file (see Table 3 below for details)
B2B characterization trending (if trend flag set to yes)
L0R/L1R ID
Work Order ID
WRS Path/Row
B2B statistics for all band combinations and SCAs

The following processing parameters that are listed in the table above can be overridden if they are given as fields within the input ODL file; correlation window size, maximum offset, minimum correlation strength, fill threshold, maximum and minimum file values.

7.2.9.5 Options

Trending on/off switch
Grid-based tie-point generation
Normalized grey scale or least squares correlation

7.2.9.6 Prototype Code

Input to the executable is an ODL file; output is a set of ASCII files containing measured offsets between band locations with and SCA.

The prototype code was compiled with the following options when creating the test data files:

```
-g -Wall -march=nocona -m32
```

The following text is a brief description of the main set of modules used within the prototype with each module listed along with a very short description. It should be noted that almost all library modules are not referenced in the explanations below. The modules within the main b2bchar directory of the

prototype are discussed and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

b2b_char
Main driver for application. Calls routines to retrieve ODL input and CPF parameters, read and verify image metadata, reduce resolution of PAN band, create a set of tiepoints, and calls module that will perform correlation on image tiepoint locations. Separate calls are made for creating tiepoints depending on whether points are to be evenly spaced or based upon a resampling grid.

get_parms
Reads input ODL parameters. Checks validity of input band combinations listed in ODL file. Reads CPF BRAA processing parameters.

verify_band_combos
Verifies search and reference band combinations given as input. Verification is done by matching reference and search band list, if bands given do match an error is returned.

create_tiepoints
Driver for creating evenly spaced tiepoints. Calls det_tiepoints for each band combination storing tiepoint locations in GCPLIB data structure.

det_tiepoints
Calculates a set of evenly spaced tiepoint locations based on image size. Tie points are based on number of points given as an input ODL parameter and the size of the image file.

create_tiepoints_grid
Driver for creating tie points based on the resampling grid.

downsample
Main driver for reducing the resolution of the PAN band. Driver calls modules to initialize reduce image file (setup_reduce_img), calculates cubic convolution weights (cubic_convolution_weights), and applies cubic convolution weights to the PAN band (reduce).

setup_reduce_img
Initializes PAN reduced image file creation.

cubic_convolution_weight
Determines cubic convolution weights.

reduce
Applies cubic weights to PAN band. Output is written to file created/initialized in setup_reduce_img.

b2b_corr
Main driver for band correlation, or band mensuration, process. Driver opens image, calls module to perform correlation at tie-point locations (process_gcp), and writes out band registration residuals file (table 1). process_gcp is called on for each SCA and band combination given in the input ODL file.

process_gcp

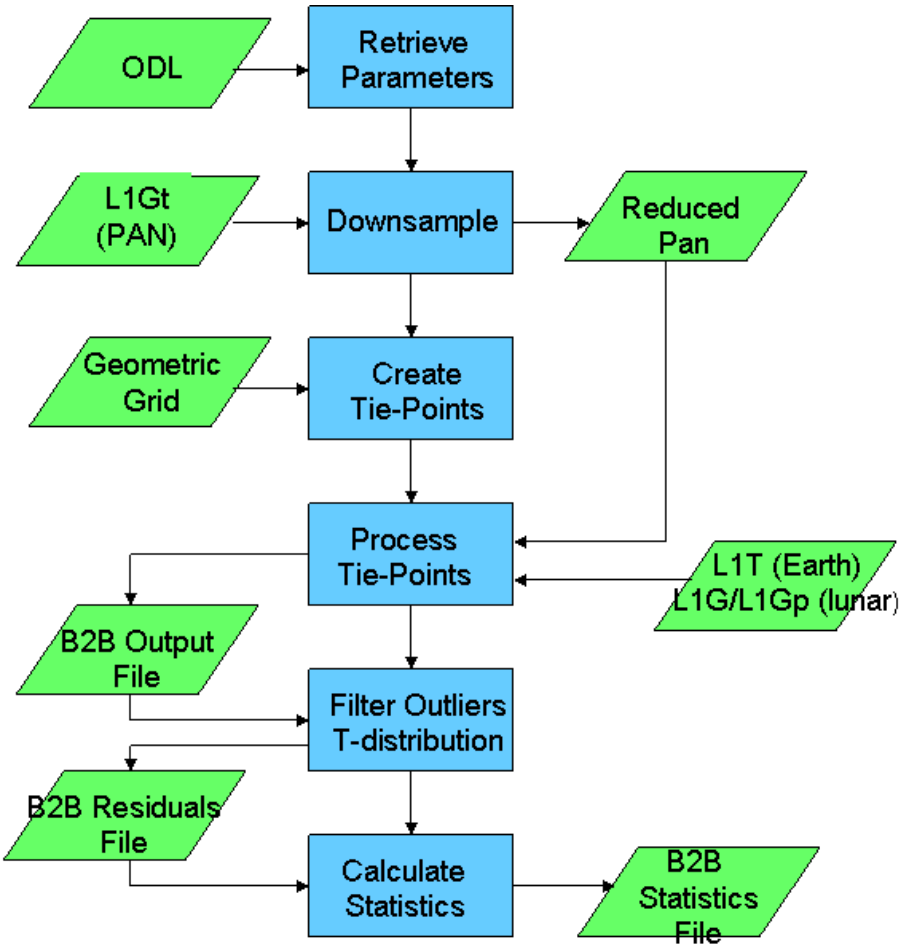
Process to perform correlation between two bands for one SCA. See Ground Control Point Correlation ADD for information on the LDCM correlation modules and process. Calls module xxx_check_fill to determine if a given window of imagery contains enough "non-fill" pixels so that mensuration can be performed.

ias_math_check_pixels_in_range
Checks to see if percent of pixels within a given buffer contains fill. Fill is passed in as a parameter. Module has been modified so that fill is a range rather than a single value.

math_fine_corr
Math library routine that implements the new (see below) least squares correlation algorithm developed for fine sub-pixel offset measurement. Takes same-size reference and search image windows as input and returns measured offsets.

7.2.9.7 Procedure

Band Registration Accuracy Assessment measures the misalignment between spectral bands after all known geometric effects have been taken into account. The results from the band registration assessment are used by the band alignment calibration routine (See Band Alignment Calibration ADD) to estimate new Legendre LOSs (See Line-of-Sight Model Creation ADD) for each band of each SCA. Due to the different viewing angles for each band of each SCA, geometric displacement due to relief must be removed from the imagery for band-to-band characterization of Earth acquisitions, i.e. input imagery for band registration assessment must be precision and terrain corrected (See Resampling ADD). The steps involved in band registration assessment are depicted in Figure 1 and include creating data sets with common pixel resolutions; choosing locations (tie-point locations) for measurement; performing mensuration; removing outliers from calculated residuals; and calculating statistics from the remaining residuals. Residuals are measured for each band combination of each SCA that is requested through the input parameters.



7.2.9.7.1

Figure 1. Band Registration Accuracy Assessment Block Diagram

The OLI Band Registration Accuracy Assessment algorithm has heritage in the Landsat 7 (L7) and Advanced Land Imagery (ALI) Image Assessments Systems (IAS) Band-to-Band Characterization (B2B Char) algorithm/process. The prototype code for OLI BRAA will contain many of the same modules that are present in the L7 and ALI IAS B2B Char. The core functions taken from ALIAS for the band-to-band assessment processes that will be needed for OLI processing are specified where applicable. Changes that may be necessary within these modules are briefly discussed. The correlation and mensuration modules however are not described within this ADD as they are already present within the Ground Control Correlation ADD, this ADD should reviewed for any information pertaining to these processes. Also it should be noted that changes due to items such as file format, which are not either instrument specific or due to changes to the algorithm, are not discussed.

7.2.9.7.2 Stage 1 - Data Input

The data input stage involves loading the information required to perform the band registration assessment. This includes reading the image file, retrieving the output B2B file names: output, residuals and statistic files; retrieving or initializing processing parameters: maximum displacement, fill range, fill threshold, minimum correlation peak, t-distribution threshold, SCAs to process, bands to process, correlation window size, trending thresholds, tie-point method; and if tie-point method is set to grid-based the geometric grid file name will be read. Once the input file, and if need be the geometric grid name, has been retrieved the files and the information stored within them can be opened and read.

7.2.9.7.3 Stage 2 - Creating a Reduced Resolution PAN band

Before displacement between the PAN band and the other multispectral bands can be measured the PAN band must be reduced in resolution to match that of the multispectral bands. An oversampled cubic convolution function is used to reduce the resolution of the PAN band. Cubic convolution interpolation uses a set of piecewise cubic spline interpolating polynomials. The polynomials have the following form:

$$f(x) = \begin{cases} (\alpha+2)|x|^3 - (\alpha+3)|x|^2 + 1 & 0 \le |x| < 1 \\ \alpha|x|^3 - 5\alpha|x|^2 + 8\alpha|x| - 4\alpha & 1 \le |x| < 2 \\ 0 & |x| > 2 \end{cases}$$

Since the cubic convolution function is a separable function, a two dimensional representation of the function is given by multiplying two one-dimension cubic convolution functions, one function representing the x-direction the other function representing the y-direction. For an offset of zero, or x = 0, and α = -1.0 the discrete cubic function has the following values; f(0) = 1 and f(n) = 0 elsewhere. Thus convolving the cubic convolution function of x = 0 with a data set leaves the data set unchanged.

$$y[n] = f[n] \otimes x[n]$$

$$\text{for } x = 0$$

$$\text{gives } y[n] = x[n]$$

$$\text{where } \otimes \text{ is the convolution operator}$$

Figure 2 shows what the cubic function f(t) (dashed line) and the corresponding discrete weights for an offset, or phase, of zero (crossed-dots).

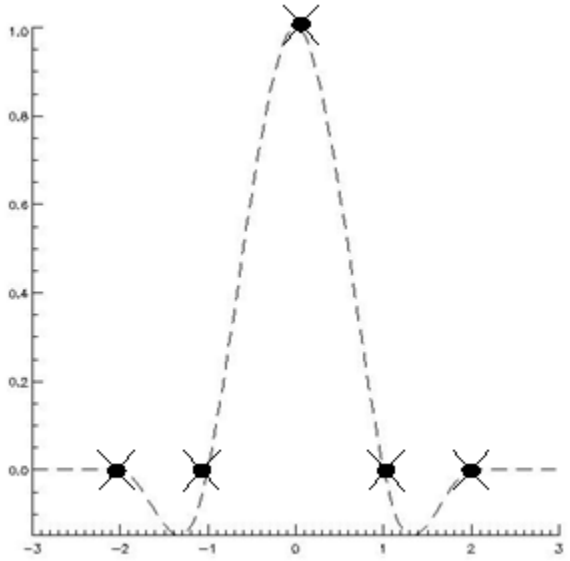


Figure 2. Cubic Convolution Function and Weights for phase of zero.

To spatially scale an input data stream an oversampled cubic convolution function with a offset of x =0 can be used. This can best be understood by looking at the Fourier Transform scaling property of a function that is convolved with a given input data set:

$$f(t) \otimes x(t) \Leftrightarrow F(\omega) \bullet X(\omega)$$

$$x(at) = \frac{1}{|a|} X\left(\frac{\omega}{a}\right)$$

Where:

$\otimes$ is convolution
● is multiplication
F is the Fourier transform of *f*
X is the Fourier transform of *x*
t is time
ω is frequency

Applying the cubic function and scaling properties to an image data file shows that densifying the points applied with the cubic convolution function will in turn inversely scale the function in the frequency domain, thus reducing the resolution of the imagery. By setting the cubic convolution offset to zero, densifying the number of weights of the cubic function, and convolving these weights to an image file a reduction in resolution will be the resultant output image file. Figure 3 shows the cubic function with corresponding weights densified by a factor of two and a phase shift of zero. To ensure that the cubic weights do not scale the DNs of the output imagery during convolution the cubic weights are divided by the scale factor.

$$f_s[n] = \frac{1}{2} \sum_{n=-4}^{4} f\left(\frac{n}{2}\right)$$

Where:

f_s[n] = scaled cubic convolution weights
f(n) = cubic convolution function

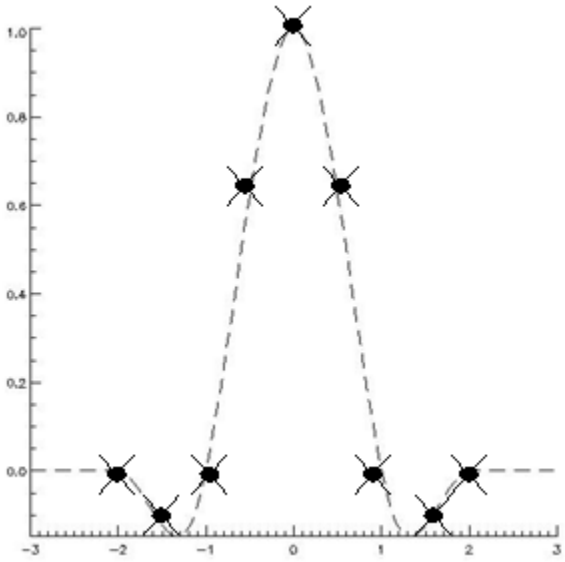


Figure 3. Cubic Convolution Densifyied by a factor of 2

Scaling the cubic convolution function by a factor of 2 gives the following 1-dimensional set of weights:

$$ccw[n] = \begin{bmatrix} 0.0 & -0.0625 & 0.0 & 0.3125 & 0.5 & 0.3125 & 0.0 & -0.0625 & 0.0 \end{bmatrix}$$

To determine the 2-dimensional cubic convolution weights two 1-dimensional sets of cubic weights are multiplied together (note only 7 values are needed for ccw, outside of this extent the weights are zero):

$$ccw[n] \times ccw[m] = ccw[n,m] = \begin{bmatrix} 0.0039 & 0.0 & -0.0195 & -0.0313 & -0.0195 & 0.0 & 0.0039 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.0195 & 0.0 & 0.0977 & 0.1563 & 0.0977 & 0.0 & -0.0195 \\ -0.0313 & 0.0 & 0.1563 & 0.25 & 0.1563 & 0.0 & -0.0313 \\ -0.0195 & 0.0 & 0.0977 & 0.1563 & 0.0977 & 0.0 & -0.0195 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0039 & 0.0 & -0.0195 & -0.0313 & -0.0195 & 0.0 & 0.0039 \end{bmatrix}$$

Where:
ccw[n] is a 8x1 1-dimensional set of cubic weights
ccw[m] is a 1x8 1-dimensional set of cubic weights

7.2.9.7.3.1 Procedure for Reducing PAN band

To reduce the resolution of the PAN band apply the ccw[n,m] weights to the PAN image data:

$$\text{reduced pan} = ccw[n,m] * \text{pan band}$$

Note: number of lines and number of samples listed below pertain to the size of the PAN band imagery.

Reduce PAN Band Resolution Processing Steps

1. Set line =0 then for every other PAN line

    1.1. Set sample = 0 then for every other PAN sample

    1.2. initialize summing variable sum = 0.0

    1.3. For m = -4 to 4

        1.3.1. For n = -4 to 4

        1.3.2. Check to see if current image index is within valid imagery

        1.3.3. if m + line < 0 then line index = -m - line
        else if m + line >= number of lines then line index =
        2 * number of lines - m - line - 1
        else line index = m + line

        1.3.4. if n + sample < 0 then sample index = -n - sample
        else if n + sample >= number of sample then sample index =
        2 * number of samples - n - sample - 1
        else sample index = n + sample

        1.3.5. sum = sum + ccw[n+4,m+4] • pan[line index, sample index]

    1.4. Store output DN for reduced PAN
    output line = line / 2
    output sample = sample / 2
    reduce pan[output line, output sample] = sum

7.2.9.7.4 Stage 3 - Create Tie-point Locations

Tie point locations may be determined in an evenly spaced pattern in output space or they may be established in an evenly spaced pattern in input space, using the OLI geometric grid.

7.2.9.7.4.1 Determine Evenly Spaced Tie-points (See notes #6 and #7)

To determine evenly spaced tie-point locations a tie-point location is defined by stepping through the output space of the imagery by the user defined steps N,M.

Create Evenly Spaced Tie-Points Processing Steps

1. Determine number of tie-points in sample and line direction:

$$\text{tie - point spacing x} = \frac{\text{ONS} - \text{correlation window samples}}{\text{M - 1}}$$

$$\text{tie - point spacing y} = \frac{\text{ONL} - \text{correlation window lines}}{\text{N - 1}}$$

Where:

M = user entered number of tie-points in sample direction
N = user entered number of tie-points in line direction
ONS = number of samples in output space of multispectral band
ONL = number of lines in output space of multispectral band
Correlation window samples = user entered correlation window size in samples
Correlation window lines = user entered correlation window size in lines

2. Set evenly spaced tie-point locations
   2.1. For j = 0 to N-2

$$\text{tie - point location y}[\text{j}] = \frac{\text{correlation window lines}}{2} + \text{j} * \text{tie - point spacing y}$$

   2.2. $$\text{tie - point location y}[\text{N} - 1] = ONL - \frac{\text{correlation window lines}}{2}$$

   2.3. For i = 0 to M-2

$$\text{tie - point location x}[\text{i}] = \frac{\text{correlation window samples}}{2} + \text{i} * \text{tie - point spacing x}$$

   2.4. $$\text{tie - point location x}[\text{M - 1}] = ONS - \frac{\text{correlation window samples}}{2}$$

7.2.9.7.4.2 Determine Geometric Grid Spaced Tie-points (See notes #6 and #7)

For descriptions of the format and data stored within the geometric grid see the Line of Sight Projection to Ellipsoid and Terrain ADD.

Geometric Space Tie-points Processing steps.

1. Read image extent parameters from geometric grid
   INS = input (raw) space number of samples
   INL = input (raw) space number of lines

2. Determine number of tie-points in sample and line direction:

$$\text{spacing x} = \frac{\text{INS} - \text{correlation window samples}}{\text{M - 1}}$$

$$\text{spacing y} = \frac{\text{INL} - \text{correlation window lines}}{\text{N - 1}}$$

3. Establish input (raw) space tie-point locations

   3.1 For j = 0 to N-2

$$\text{y}[\text{j}] = \frac{\text{correlation window lines}}{2} + \text{j} * \text{spacing y}$$

3.2 $y[N-1] = INL - \frac{\text{correlation window lines}}{2}$

3.3 For i = 0 to M-2

$$x[i] = \frac{\text{correlation window samples}}{2} + i * \text{spacing x}$$

3.4 $x[M-1] = INS - \frac{\text{correlation window samples}}{2}$

4. Project inputs space tie-points locations to output space

4.1 For j=N-1

4.1.1 For i=M-1

Map input space tie-point location to output space using grid mapping coeffcients.

tie-point location y = $b_0 + b_1 * x[i] + b_2 * y[j] + b_3 * x[i] * y[j]$

tie-point location x = $a_0 + a_1 * x[i] + a_2 * y[j] + a_3 * x[i] * y[j]$

Where (See note #7):

a_n = geometric grid forward sample mapping coefficients for zero elevation plane retrieved from the resampling grid

b_n = geometric grid forward line mapping coefficients for zero elevation plane retrieved from the resampling grid

7.2.9.7.5 Stage 4. Calculate Individual Point-by-Point Band Displacements

Normalized cross correlation is the standard method used to measure spatial differences between the reference and search windows extracted from the two bands being compared. The normalized cross correlation process helps to reduce any correlation artifacts that may arise from radiometric differences between the two image sources. The correlation process will only measure linear distortions over the windowed areas. By choosing appropriate correlation windows that are well distributed throughout the imagery, nonlinear differences between the image sources can be found. Normalized grey scale correlation has the following formula:

$$R(x,y) = \frac{\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2}\left[\left(f(j,i)-\bar{f}\right)\left(g(x+j,y+i)-\bar{g}\right)\right]}{\left[\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2}\left(f(j,i)-\bar{f}\right)^2 \sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2}\left(g(x+j,y+i)-\bar{g}\right)^2\right]^{1/2}}$$

Where:

N = M = Correlation window size in lines and samples

R = correlation surface (N,M) (See note# 10)

F = reference window (N,M)

G = search window (N,M)

$$\bar{f} = \frac{1}{(M+1)(N+1)} \sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2} f(j,i)$$

$$\bar{g} = \frac{1}{(M+1)(N+1)} \sum_{j=-N/2}^{N/2} \sum_{i=-M/2}^{M/2} g(x+j, y+i)$$

Normalized cross correlation will produce a discrete correlation surface (i.e., correlation values at integer x,y locations). A sub pixel location associated with the displacement is found by fitting a polynomial around a 3x3 area centered on the correlation peak. The polynomial coefficients can be used to solve for the peak or sub pixel location. Once the discrete correlation has been calculated and the peak value within these discrete values has been found the sub-pixel location can be calculated:

$$P(y,x) = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 + a_5 y^2$$

Where
P(x,y) is polynomial peak fit
x = sample direction
y = line direction

Set up matrices for least squares fit of discrete R(x,y) to x/y locations.

$$\underset{9x1}{\begin{bmatrix} R(-1,-1) \\ R(-1,0) \\ \vdots \\ R(1,0) \\ R(1,1) \end{bmatrix}} = \underset{9x6}{\begin{bmatrix} 1 & -1 & -1 & (-1)*(-1) & (-1)^2 & (-1)^2 \\ 1 & 0 & -1 & (-1)*(0) & (0)^2 & (-1)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 1 & (1)*(0) & (0)^2 & (0)^2 \\ 1 & 1 & 1 & (1)*(1) & (1)^2 & (1)^2 \end{bmatrix}} \underset{6x1}{[a]}$$

or: [Y] = [X] [a]

Note that R(x,y) is relative to the peak, the total offset will need to have the integer line offset and sample offset added to the sub-pixel location to have the total measured offset. Solving for the peak polynomial using least squares:

$$[a] = \left([X]^T [X] \right)^{-1} [X]^T [Y]$$

Calculating the partial derivative of P(x,y) in both the x and y directions, setting the partial equations to zero, and solving the partials for x and y, gives the sub-pixel location within the sub-pixel 3x3 window.

$$\frac{\partial}{\partial x} P(x,y) = a_1 + a_3 y + 2a_4 x = 0$$
$$\frac{\partial}{\partial y} P(x,y) = a_2 + a_3 x + 2a_5 y = 0$$

Set partial equations equal to zero and solve for x and y:

$$\text{Sub-pixel x offset} = \frac{2a_1 a_5 - a_2 a_3}{a_3^2 - 4a_4 a_5}$$

$$\text{Sub-pixel y offset} = \frac{2a_2a_4 - a_1a_3}{a_3^2 - 4a_4a_5}$$

The steps for mensuration, calculating the total offset measured, and how they fit in the overall procedure is given in the processing steps section.

See the Ground Control Point Correlation ADD for prototype specifications of the normalized grey scale correlation processes.

Least Squares Fine Correlation Method
The band-to-band and image-to-image accuracy characterization algorithms also provide a second, least-squares based correlation method that can be used to measure sub-pixel image displacements somewhat more reliably than the cross-correlation/peak finding method used for general purpose correlation. This is useful for band registration measurements where the displacements should always be much less than a pixel, and where the quadratic peak finding method can introduce small offset-dependent biases in the measurements. This method requires that the reference and search image windows be the same size and that the offsets to be determined be less than 1 pixel. Since the normalized grey scale correlation algorithm does not work on image windows whose dimensions are not even numbers, this least squares correlation method is invoked if either window dimension is an odd number.

The least squares correlation method uses the reference and search image window pixels to estimate the sample offset (∆sample), line offset (∆line), gain, and bias adjustments that best match the (∆sample, ∆line) shifted and bilinearly interpolated search image to the radiometrically adjusted (1+∆gain, ∆bias) reference image. The 3x3 pixel image sub-window surrounding each interior (non-edge) image pixel in the reference and search windows provides one observation for purposes of estimating the four adjustment parameters, using the following model:

$$S_0 + S_x * \Delta\text{sample} + S_y * \Delta\text{line} + S_{xy} * \Delta\text{sample} * \Delta\text{line} = R_0 * (1+\Delta\text{gain}) - \Delta\text{bias}$$

Where:
$S_0 = S_{i,j}$ = the central pixel in the 3x3 search sub-window centered at (i,j)
$S_x = (S_{i+1,j} – S_{i-1,j})/2$ = slope estimate in the sample direction
$S_y = (S_{i,j+1} – S_{i,j-1})/2$ = slope estimate in the line direction
$S_{xy} = (S_{i+1,j+1} + S_{i-1,j-1} – S_{i+1,j-1} - S_{i-1,j+1})/4$ = rate of slope change
$R_0 = R_{i,j}$ = the reference image pixel corresponding to $S_{i,j}$

This can be reorganized into an observation model for the 4 fit parameters:

$$S_x*\Delta\text{sample} + S_y*\Delta\text{line} - R_0*\Delta\text{gain} + \Delta\text{bias} = R_0 - S_0 - S_{xy}*\Delta\text{sample}*\Delta\text{line}$$

Or:

$$\begin{bmatrix} S_x & S_y & -R_0 & 1 \end{bmatrix} \begin{bmatrix} \Delta sample \\ \Delta line \\ \Delta gain \\ \Delta bias \end{bmatrix} = \begin{bmatrix} R_0 - S_0 - S_{xy}\,\Delta sample \Delta line \end{bmatrix}$$

[1 x 4] [4 x 1] = [1 x 1] Array sizes

Note that this equation is not linear (since Δsample and Δline appear on the right hand side) and must be solved iteratively.

Each non-edge pixel generates an observation of this form:

$$[X_{i,j}]^T\,[coef] = [Y_{i,j}]$$

Where:

$[X_{i,j}]^T = [\ S_x\ \ S_y\ \ -R_0\ \ 1\ ]$ (1x4 matrix)
$[coef] = [\ \Delta sample\ \ \Delta line\ \ \Delta gain\ \ \Delta bias\ ]^T$ (4x1 matrix)
$[Y_{i,j}] = [\ R_0 - S_0 - S_{xy}\,\Delta sample\ \Delta line\ ]$ (1x1 matrix)

Taken together, these observations can be used to compute the best fit, in the least squares sense, values for the four fit parameters:

$[N] = \Sigma\,[X_{i,j}]\,[X_{i,j}]^T$ (4x4 matrix)
$[C] = \Sigma\,[X_{i,j}]\,[Y_{i,j}]^T$ (4x1 matrix)
$[coef] = [N]^{-1}\,[C]$ (4x1 matrix)

The computed values of the fit parameters in [coef] are used to update the $[Y_{i,j}]$ values for each iteration.

The solution procedure is as follows:

1. Verify that the input reference and search windows are the same size and that the window dimensions are both at least 3 pixels.
2. Initialize the least squares solution normal equations:
    a. Set all 4 elements of the 4x1 constants vector **C** to zero.
    b. Set all 16 elements of the 4x4 normal equation matrix **N** to zero.
    c. Set the normal equation diagonal term corresponding to the gain parameter, **N**[2][2], to $1/\sigma_g^2$, where σ_g is the apriori standard deviation of the gain parameter, set to 0.05 (5%) to limit the magnitude of the gain adjustment.
    d. Set the normal equation diagonal term corresponding to the bias parameter, **N**[3][3], to $1/\sigma_b^2$, where σ_b is the apriori standard deviation of the bias parameter, set to 5 DN to limit the magnitude of the bias adjustment.
    e. Initialize the four adjustment parameter values to zero.
3. Iterate the solution 3 times. For each iteration:
    a. Loop through the non-edge image pixels, $S_{i,j}$, $R_{i,j}$, in the N_{samp} by N_{line} image windows, where $0 < i < N_{samp}-1$ and $0 < j < N_{line}-1$. For each pixel:
        i. Compute S_0, S_x, S_y, and S_{xy} from the 3x3 search sub-window surrounding the current pixel using the equations above.
        ii. Compute the right hand side of the observation equation using R_0 and the current estimates of Δsample and Δline:
            $RHS = R_0 - S_0 - S_{xy} * \Delta sample * \Delta line$
        iii. Add this observation to the normal equations:
            N[0][0] += $S_x * S_x$
            N[0][1] += $S_x * S_y$
            N[0][2] -= $S_x * R_0$

N[0][3] += S_x
C[0] += S_x * RHS
N[1][1] += S_y * S_y
N[1][2] -= S_y * R_0
N[1][3] += S_y
C[1] += S_y * RHS
N[2][2] += R_0 * R_0
N[2][3] -= R_0
C[2] -= R_0 * RHS
N[3][3] += 1
C[3] += RHS

b. Complete the symmetric normal equation matrix:
N[1][0] = **N**[0][1]
N[2][0] = **N**[0][2]
N[2][1] = **N**[1][2]
N[3][0] = **N**[0][3]
N[3][1] = **N**[1][3]
N[3][2] = **N**[2][3]

c. Solve the normal equations:
X = [Δsample Δline Δgain Δbias]^T = **N**$^{-1}$ **C**

4. Return the results of the final iteration:
Fit_offset[0] = Δsample
Fit_offset[1] = Δline
Diag_Displacement = sqrt(Δsample * Δsample + Δline * Δline)

7.2.9.7.6 Stage 5. Removing Outliers Using the t-distribution

Once all the line and sample offsets have been measured and the first level of outlier rejection has been performed, a check against the maximum allowable offset and the minimum allowable correlation peak, the measurements are further reduced of outliers using a Student-t outlier rejection.

Given a t-distribution tolerance value, outliers are removed within the data set until all values deemed as “non-outliers” or “valid” fall inside the confidence interval of a t-distribution. The tolerance, or associated confidence interval, is specified per run (or processing flow) and usually lies between 0.9-0.99. The default value is 0.95. The number of degrees of freedom of the data set is equal to the number of valid data points minus one. The steps involved in this outlier procedure are given below. The process listed works on lines and samples simultaneously, calculating statistics independently for each.

Student-t Outlier Rejection Processing steps.

1. Calculate mean and standard deviation of data for lines and samples (see stage #6).

$$\text{mean offset} = \frac{1}{N}\sum_{i=0}^{N}\text{offset}_i$$

$$\text{standard deviation} = \frac{1}{N-1}\sum_{i=0}^{N}\left(\text{offset}_i - \text{mean offset}\right)^2$$

Where:
N = number of valid offsets measured (above peak threshold and below maximum offset)

Two means and standard deviations are calculated, one for the line direction and one for the sample direction.

2. Find largest offset and compare it to outlier threshold.

   2.1. Calculate two tailed t-distribution (T) value for current degree of freedom (N-1) and confidence level α.

   2.2. Calculate largest deviation from the mean allowable for the specified degree of freedom and α:
   $\Delta line = \sigma_{line} * T$
   $\Delta sample = \sigma_{sample} * T$
   Where:
   σ_{line} = standard deviation of valid line offsets
   σ_{sample} = standard deviation of valid sample offsets

   2.3. Find valid data point that is farthest from the mean.
   $\max line_i = MAX\{$ line offset - mean line offset$\}$
   $\max sample_j = MAX\{$ sample offset - mean sample offset$\}$
   Where:
   The maximum is found from all valid offsets
   i is the tie-point number of max line
   j is the tie-point number of max sample

   2.4. If valid data point that is farthest from the mean is greater than the allowable Δ then the valid point is flagged as outlier.
   if $\max line_i > \Delta line$ or $\max sample_j > \Delta sample$ then
   if($\max sample_j / \sigma_{sample} > \max line_i / \sigma_{line}$)
   tie-point j is marked as an outlier
   else
   tie-point i is marked as an outlier
   else no outliers found

3. Repeat 1 and 2 above until no outliers are found.

7.2.9.7.7 Stage 6. Calculating Measured Statistics

The mean, standard deviation, minimum, maximum, median, and root-mean squared offset (RMS) are calculated from the tie-points that pass all outlier criteria; below maximum offset, above peak threshold, and student t-distribution test. The calculation for mean, standard deviation, and RMS are shown below where x_i is the measured offset.

mean: $m_x = \frac{\sum_{i=0}^{N-1} x_i}{N}$

standard deviation: $\sigma_x = \sqrt{\frac{\sum_{i=0}^{N-1}(x_i - m_x)^2}{N-1}}$

RMS: $RMS_x = \sqrt{\frac{\sum_{i=0}^{N-1}(x_i)^2}{N}}$

7.2.9.7.7.1 Band Accuracy Assessment Processing steps

Windows extracted from imagery have the user entered dimensions; correlation window lines and correlation window samples. Correlation parameters have been read or set as default values; maximum offset, fit method, correlation peak, fill data range, fill threshold. The bands should be indexed so that the PAN band is always used as a reference to all other bands.

1. For SCA = Number of SCAs to process

   1.1. For rband = Number of OLI bands to process

      if rband is equal to PAN use reduced PAN image file

   1.2. For sband = rband + 1 to Number of OLI bands to process

   1.3. For index = Number of tie-points to process

      1.3.1. Read current tie-point chip and tie-point location x,y
      Set tie-point flag to unsuccessful

      1.3.2. Extract sband window (of imagery) at tie-point location x,y

      1.3.3. Extract rband window (of imagery) at tie-point location x,y

      1.3.4. Count number of pixels in rband window that is within fill range.
      count = 0
      For i=0 to number of pixels in correlation window
      If rband pixel is > fill min and rband pixel is < fill max
      count++

      1.3.5. Check number of rband pixels counted against fill threshold/percentage.
      if $\frac{\text{count}}{\text{correlation window size}} > \text{fill threshold}$
      increment index to next tie-point location
      else
      continue

      1.3.6. Count number of pixels in sband window that is within fill range.
      count = 0

For i=0 to number of pixels in correlation window
If sband pixel is > fill min and sand pixel is < fill max
count++

1.3.7. Check number of sbands pixels counted against fill threshold/precentage.

$$\text{if } \frac{\text{count}}{\text{correlation window size}} > \text{fill threshold}$$

increment index to next tie-point location
else
continue

1.3.8. Perform normalized grey scaled correlation between rband and sband windowed images, calculating correlation surface *R* (See Stage 4 and notes #9 and #10).

1.3.9. Find peak within correlation surface
Max = *R*(0,0)
For i=0 to correlation window number of lines -1
For j=0 to correlation window number of samples -1
If *R*(i,j) > max then
Max = *R*(i,j)
line offset = i
sample offset = j

1.3.10. Check correlation peak against threshold
if max > peak threshold
continue
else
set tie-point flag to outlier and choose next tie-point

1.3.11. Measure sub-pixel peak location (see stage #4)
Δsub-line
Δsub-sample

1.3.12. Calculate total pixel offset
total line offset = line offset + Δsub-line
total sample offset = sample offset + Δsub-sample

1.3.13. Check offset against maximum displacement offset

$$\text{total displacement} = \sqrt{(\text{total line offset})^2 + (\text{total sample offset})^2}$$

if (total displacement > maximum displacement)
Set tie-point flag to outlier and choose next tie-point
Else
Set tie-point flag to valid

1.4. Store SCA and band combination (rband-to-sband) tie-point mensuration information, correlation success, and offsets measured. See table #1.

2. For SCA = 1 to Number of SCAs to process

2.1. For band combination = 1 to Number of band combinations

2.1.1. Perform t-distribution outlier rejection (See stage #5).

2.2. Store SCA and band combination final individual tie-point information and outlier flag. See table #2.

3. For SCA = 1 to Number of SCAs to process

4. For band combination = 1 to Number of band combinations

4.1. Calculate mean, minimum, maximum, median, standard deviation, and root mean squared offset.

4.2. Store SCA and band combination statistics. See table #3.

5. Perform trending if trending flag is set to yes

5.1. Check results against trending thresholds
For each band of each SCA
if measured Standard Deviation > trending threshold
exit trending
If there are no Standard Deviation > trending thresholds perform trending

7.2.9.8 Output files

The output files listed below for the BRAA follow the philosophy of the Advanced Land Imagery Image Assessment System (ALIAS) Band-to-Band (B2B) Characterization output files in that they are made generic so that the same format can be used elsewhere. Therefore some fields like latitude, longitude, and elevation may not apply to the application and would be filled with zeros or nominal values.

All output files contain a standard header. This standard header is at the beginning of the file and contains the following:

1) Date and time file was created.
2) Spacecraft and instrument pertaining to measurements.
3) Off-nadir (roll) angle of spacecraft/instrument.
4) Acquisition type
5) Report type (band-to-band)
6) Work order ID of process (left blank if not applicable)
7) WRS path/row
8) Software version that produced report.
9) L0R image file name

The data shown within Table 3 listed below is stored in the database. The statistics stored per band per SCA will be used for trending analysis of the band registration accuracy of the OLI instrument. Results produced through a time-series analysis of this data stored, over a set time interval or

multiple image files, will determine if new Line-of-Sight (LOS) Legendre coefficients will need to be generated from the OLI Band-to-Band Calibration Algorithm (See OLI Band-to-Band Calibration ADD for details). These statistics may also be needed for providing feedback to the LDCM user community about the band registration of LDCM products generated.

281. Field	**282. Description**
283. Date and time	284. Date (day of week, month, day of month, year) and time of file creation.
285. Spacecraft and instrument source	286. LDCM and OLI (TIRS if applicable)
287. Processing Center	288. EROS Data Center SVT
289. Work order ID	290. Work order ID associated with processing (blank if not applicable)
291. WRS path/row	292. WRS path and row (See note #11)
293. Software version	294. Software version used to create report
295. Off-nadir angle	296. Off-nadir roll angle of processed image file (See note #12)
297. Acquisition Type	298. Earth viewing or Lunar
299. L0R image file	300. L0R image file name used to create L1T
301. Processed image file name	302. Name of L1T used to create report
303. Reference bands	304. Reference bands used in band assessment
305. Search bands	306. Search bands used in band assessment
307. Heading for individual tie-points	308. One line of ASCII text defining individual tie-point fields.
309. For each tie-point:	310.
311. Tie point number	312. Tie-point index/number in total tie-point list
313. Reference line	314. Tie-point line location in reference image (band)
315. Reference sample	316. Tie-point sample location in reference image (band)
317. Reference latitude	318. Tie-point latitude location
319. Reference longitude	320. Tie-point longitude location
321. Reference elevation	322. Elevation of tie-point location (see note #13)
323. Search line	324. Tie-point line location in search image
325. Search sample	326. Tie-point sample location in search image
327. Delta line	328. Measured offset in line direction
329. Delta sample	330. Measured offset in sample direction
331. Outlier flag	332. 1=Valid, 0=Outlier
333. Reference band	334. Reference band number
335. Search band	336. Search band number
337. Reference SCA	338. SCA number that reference window was extracted
339. Search SCA	340. SCA number that search window was extracted
341. Search image	342. Name of search image
343. Reference image	344. Name of reference image

Table 6. Band Registration Accuracy Assessment Data File

345. Field	**346. Description**
347. Date and time	348. Date (day of week, month, day of month, year) and time of file creation.
349. Spacecraft and instrument source	350. LDCM and OLI (TIRS if applicable)
351. Processing Center	352. EROS Data Center SVT
353. Work order ID	354. Work order ID associated with processing (blank if not applicable)
355. WRS path/row	356. WRS path and row (See note #11)
357. Software version	358. Software version used to create report
359. Off-nadir angle	360. Off-nadir pointing angle of processed image file (See note #12)
361. Acquisition Type	362. Earth viewing or Lunar
363. L0R image file	364. L0R image file name used to create L1T
365. Processed image file name	366. Name of L1T used to create report
367. Number of records	368. Total number of tie-points stored in file
369. Heading for individual tie-points	370. One line of ASCII text defining individual tie-point fields.
371. For each band combination	372.
373. Combination header	374. Number of points in combination, reference band number, search band number.
375. For each tie-point:	376.
377. Tie point number	378. Tie-point index/number in total tie-point list
379. Reference line	380. Tie-point line location in reference image (band)
381. Reference sample	382. Tie-point sample location in reference image (band)
383. Reference latitude	384. Tie-point latitude location
385. Reference longitude	386. Tie-point longitude location
387. Reference elevation	388. Elevation of tie-point location
389. Search line	390. Tie-point line location in search image
391. Search sample	392. Tie-point sample location in search image
393. Delta line	394. Measured offset in line direction
395. Delta sample	396. Measured offset in sample direction
397. Outlier flag	398. 1=Valid, 0=Outlier

399.	Correlation coef	400.	Correlation coefficient for tie point correlation
401.	Reference band	402.	Reference band number
403.	Search band	404.	Search band number
405.	Reference SCA	406.	SCA number that reference window was extracted from
407.	Search SCA	408.	SCA number that search window was extracted from
409.	Search image	410.	Name of search image
411.	Reference image	412.	Name of reference image

Table 7. Band Registration Accuracy Assessment Residuals File

413. Field	**414. Description**
415. Date and time	416. Date (day of week, month, day of month, year) and time of file creation.
417. Spacecraft and instrument source	418. LDCM and OLI (TIRS if applicable)
419. Processing Center	420. EROS Data Center SVT
421. Work order ID	422. Work order ID associated with processing (blank if not applicable)
423. WRS path/row	424. WRS path and row (See note #12)
425. Software version	426. Software version used to create report
427. Off-nadir angle	428. Off-nadir pointing angle of processed image file (See note #13)
429. Acquisition Type	430. Earth viewing or Lunar
431. L0R image file	432. L0R image file name used to create L1T
433. Processed image file name	434. Name of L1T used to create report
435. t-distribution threshold	436. Threshold used in t-distribution outlier rejection
437. For each band combination of each SCA processed	438.
439. Reference band	440. Reference band of statistics
441. Search band	442. Search band of statistics
443. SCA	444. SCA number of statistics
445. Total tie-points	446. Total number of tie-points for band combination of SCA
447. Correlated tie-points	448. Number of tie-points that successfully correlated for band combination of SCA
449. Valid tie-points	450. Total number of valid tie-points for band combination of SCA after all outlier rejection has been performed
451. For both line and sample direction:	452. All statistics are given in terms of pixels
453. Minimum offset	454. Minimum offset within all valid offsets
455. Mean offset	456. Mean offset of all valid offsets
457. Maximum offset	458. Maximum offset within all valid offsets
459. Median offset	460. Median offset within all valid offsets
461. Standard deviation	462. Standard deviation of all valid offsets
463. Root-mean-squared	464. Root mean squared offset of all valid offsets

Table 8. Band Registration Accuracy Assessment Statistics Output File

7.2.9.8.1 Assessing Band Registration (Accessing Statistics Stored in Database)

The Band Accuracy Assessment statistics stored in the database will need to be accessed by the geometric CalVal team. Delineation, or essentially data base querying, will be done by the following or a combination of the following:

1) Date range of image acquisition or processing date
2) By SCA number
3) By band number
4) By acquisition type (Nadir, off-nadir, Lunar)
5) By geographic location of image extent.

At a minimum access to the Band Accuracy Assessment statistics is needed. Simple tools, such as an SQL queries, would be beneficial to the geometric CalVal team but are not absolutely necessary as they could be developed later through other means.

7.2.9.9 Maturity

7.2.9.10 Notes

Some additional background assumptions and notes include:

18. Correlation parameters, minimum correlation peak and maximum offset, are stored and retrieved from the CPF.
19. Options need to be available for generating statistics; scene statistics, individual bands per SCA, SCA summary, band summary. These statistics would be provided to the user as summary statistics to be provided as image quality assessment to the user community.
20. There will need to be a set of criteria, based on calculated statistics, as to whether trending should be performed or not. These criteria would be provided to avoid having garbage stored in the database. Any values needed in determining whether the criteria have been met for trending would be stored and retrieved from the CPF. There would be one threshold per band per SCA. The criteria to check for trending are shown in section 5.1 of the Band Accuracy Assessment Processing steps section.
21. Band Accuracy statistics stored within the database will be accessed for analysis.
    a. *Accessed according to a specific date range.*
    b. *Accessed according to a specific band or SCA.*
    c. *Accessed according to a specific geographic location.*
    d. *Accessed according to acquisition type (nadir, off-nadir, lunar).*

    This data accessed will be retrieved and stored within a comma delimited file. The methodology used to access the database could be an SQL script. This SQL query code could be developed either by the IPE or outside of the IPE.
22. Data stored within the database will be accessed for time series analysis.
    a. Data would be pulled out by scene/SCA band pairs for a user-specified time period.
    b. Statistics over multiple scenes would be calculated per SCA and/or per band. Then combined them into the SCA and/or band average statistics.

c. Results could be compared to the band registration spec. These results could serve as triggers to other events, i.e. new CPF generation and testing.
d. Results could be used to verify conformance with product specifications.

These calculations could be performed within the methodology used to access the data from the database (SQL script).

23. Tie-point locations could also be stored and used as projection Y and X coordinates. The appropriate conversions must be done when converting between projection coordinates and line and sample locations when extracting image windows between bands. This transformation should also include any rotation due to path orientated projections.
24. The prototype code uses a library call that maps any input point with a given elevation to output space. For BRAA, the elevation for the mapping point is set to zero. Since for BRAA the reference and search output space are the same, output line/sample in output reference space should be line/sample in output search space.
25. The c and d parallax coefficients are needed for each band or each SCA for every grid cell point. Therefore if the coefficients were stored as arrays stacked by grid column and then grid row for a particular input pixel that fell within grid cell column N and grid cell row M the c and d coefficients needed for that pixel would be indexed by: index = (M * number of grid columns + N) * 2. The factor of 2 is due to the fact the parallax odd/even effects are mapped as linear therefore 2 coefficients are stored for each the odd and even pixels of a grid cell.
26. The grey scale correlation process, or surface, can be implemented using a Fast Fourier Transform (FFT).
27. The correlation surface could be smaller than the search window depending on the search area or maximum offset.
28. Any kind of "non-WRS" collect; like lunar, should have 000/000 listed as the path/row.
29. Pointing angle for lunar acquisitions would be 0.0.
30. This tie point residual file structure is also used for the image registration accuracy characterization algorithm so it includes fields that are not required for both algorithms. An example is the elevation field which is set to 0 for this algorithm.
31. The correlation result fit method defines the algorithm used to estimate the correlation peak location to sub-pixel accuracy. Only the quadratic surface fitting method described in this ADD is supported in the baseline algorithm. The Least-Squares Correlation technique does not use the surface fitting method, for the grey scale correlation technique the peak fitting method still applies.

7.2.10 OLI Band-to-Band Calibration Algorithm

7.2.10.1 Background/Introduction

The Band-to-Band Calibration (B2BCal), or Band Calibration, algorithm estimates improved values for band placement within each Sensor Chip Assembly (SCA) of the OLI instrument. Adjustments are made relative to the PAN band, or in other words, the PAN band serves as the reference for all other bands.

The B2B calibration takes the Band Accuracy Assessment residuals file, which represents displacements with respect to the product output projection space, maps the residuals back into displacements with respect to the focal plane and then performs a least squares (LSQ) fit between the focal plane residuals to determine updates to the OLI band Legendre line-of-sight (LOS) polynomial coefficients. The least squares fit results represent updates needed to adjust the existing Legendre LOS coefficients. These updates can be used to produce new Legendre LOS coefficients for the Calibration Parameter File (CPF).

An Earth based acquisition will be used to calibrate all bands except the cirrus. A lunar acquisition or a high elevation Earth target acquisition will be used to calibrate the cirrus band.

7.2.10.2 Dependencies

The OLI B2B calibration algorithm assumes that a cloud free nadir viewing L1T image has been generated and the resampled DEM used to create the L1T is available. The Model Creation and LOS Projection/Gridding algorithms for the L1T will be assumed to have been executed and the corresponding output files available. The L1T image needs to be in a SCA separated format and either in a SOM or UTM path oriented projection. The digital orthophoto quadrangle (DOQ) control and best available digital elevation model (DEM) needs to be used in generating the L1T. The accuracy of the precision solution should have post-fit residuals below the recommended threshold, the solution should have used an adequate number of control points, and the distribution of the control should be well distributed throughout the imagery. The Band Registration Accuracy Assessment, or Band Characterization (B2BChar), algorithm will assumed to have been run on the L1T image successfully producing a Band Accuracy Assessment residuals file. This Band Accuracy Assessment residuals file, along with the CPF, geometric line of sight model, the co-registered DEM, and the geometric line of sight resampling grid, are used as inputs to the Band Calibration algorithm.

7.2.10.3 Inputs

The B2B calibration algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs	**Algorithm Inputs**
OLI resampling grid	ODL
DEM	ODL
OLI CPF file name	ODL
Along track IFOV	CPF/LOS-model
Minimum points	ODL
Number of Legendre Coefficients	ODL (See note #6)
OLI Line-of-Sight model	ODL

B2B residuals file	ODL
Band calibration report file	ODL
Trend flag	ODL
Flag for CPF group creation (see note #3)	ODL
Flag for individual tie-point listing	ODL
CPF effective dates (begin and end)	ODL
L0R ID (for trending)	ODL
Work Order ID (for trending)	ODL

7.2.10.4 Outputs

B2B calibration report file (See note #1 and table #1)
Legendre LOS CPF group
B2B calibration trending
L0R/L1R ID
Work Order ID
WRS Path/Row
B2B calibration post and pre fit residuals
New SCA line-of-sight parameters

7.2.10.5 Options

Trending on/off switch
fits LOS group within OLI CPF.

7.2.10.6 Prototype Code

Input to the executable is an ODL file, output is an ASCII file containing measured offsets between band combinations of the L1T image and the corresponding updated line-of-sight (LOS) Legendre CPF coefficients. Under this directory is the ODL input file needed, band accuracy assessment residuals file, the input CPF, the output reports file and the output updated Legendre LOS coefficients.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The following are a set of brief descriptions of the main set of modules used within the prototype. It should be noted that almost all library modules are not referenced in the explanations below. The modules within the main bandcal directory or the prototype are discussed and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

getpar
Reads the parameters from the input ODL parameter file. Input parameters include: co-registered DEM, CPF, LOS resampling grid, geometric LOS model file and output band calibration report file names, the minimum points, number of coefficients, effective CPF file dates, and output file print flags. The minimum points variable ensures that the normal matrix contains a minimum number along its diagonal to zero out any omitted bands. Rather than being removed from the solution, the offsets for omitted bands are set to zero with a weight equal to the minimum number of points. Omitted bands, for calibration or adjustment, are dependent on the bands present within the band

accuracy assessment residuals file. A similar approach is used to restrict the number of SCAs that will be calibrated.

read_b2bchout
Reads band accuracy assessment residuals file. Also determines the specific SCAs and bands to calibrate by checking the band accuracy assessment residuals for SCAs and bands present.

oli_get_dem
Reads DEM file into IMAGE data structure.

oli_get_model
Reads OLI geometric/LOS model.

oli_get_grid
Reads OLI LOS geometric resampling grid.

12. bandcal
13. Main driver for determining new Legendre LOS. Calls module to retrieve ODL parameters (getpar), calls module to read band accuracy assessment residuals files (read_b2bchout), reads elevation or DEM data (oli_get_dem), reads LOS geometric resampling grid, reads geometric line-of-sight model, and solves for new Legendre LOS (solve_focal_plane).
14. solve_focal_plane
15. Module to solve for new Legendre LOS. Loops on each valid tie-point for each SCA and each band combination. Calls module get_los_errors to determine per tie-point adjustment needed for determining least squares (LSQ) solution for new Legendre LOS coefficients. Calculates post and pre-fit statistics associated with Legendre LOS coefficients.
16. get_los_errors
17. Calculates delta input line and sample LOS needed for LSQ. Reads elevation for tie-point, maps tie-point to input space, finds adjustment needed between search and reference LOS vectors.
18. write_SCA_parameters_cpf
Writes out a new set of Legendre LOS. Format fits LOS group within OLI CPF.

7.2.10.7 Procedure

Band calibration uses the residuals measured during the Band Registration Accuracy Assessment Algorithm (See the Band Registration Accuracy Assessment ADD) to determine updates to the Legendre LOS coefficients (See Line-of-Sight Model Creation ADD). The band calibration process involves taking the residuals from band registration accuracy assessment, measured in output space, mapping them into input space angular deltas in terms of along- and across-track LOS angles and performing a least squares fit of the input space LOS angle deltas to a set of 2^{nd} order Legendre polynomial correction coefficients. The correction polynomials calculated represent updates to the original LOS Legendre polynomial coefficients. New Legendre LOS coefficients can be found by combining the correction coefficients with the original coefficients.

Due to the differences in viewing geometry between bands within a SCA, along with the differences in viewing geometry between SCAs, the effects due to relief displacement must be taken into account during band calibration. To account for relief displacement during B2B calibration a DEM is required. The resampling grid and LOS model is also required during B2B calibration. The resampling grid, the corresponding detector's IFOV, and the LOS model's Legendre coefficients are used to map the residuals from output space to angular differences in input space.

A least squares fit is done on all requested bands and SCAs using the band-to-band tie point measurements from all band-pair combinations for a single SCA at a time. Requested bands and SCAs to process are based on the bands and SCAs present within the Band Registration Accuracy Assessment residuals file.

7.2.10.7.1 Stage 1- Data input

The data input stage involves loading the information required to perform the band calibration. Input file names are needed for: geometric LOS resampling grid, LOS model, band registration accuracy assessment results (B2B residuals file), output band calibration report file name, and the L1T DEM file name. Further input parameters are the effective begin and end dates of the new Legendre LOSs calculated, trending flag, CPF group creation flag, and individual tie-point reporting. Once the file names for the input data needed are retrieved the files can be opened and read.

Get ODL Parameters
Reads the parameters from the input ODL parameter file. This process was modified from the ALIAS heritage version to handle new inputs: minimum points, flag for CPF group creation, CPF effective dates, and flag for reporting individual tie-point results. The minimum points variable ensures that the normal matrix contains a minimum number along its diagonal to zero out any omitted bands. Rather than being removed from the solution, the offsets for omitted bands are set to zero with a weight equal to the minimum number of points.

Read Band-to-Band Residual File
Reads band accuracy assessment residuals file.

Read DEM
Read DEM file into IMAGE data structure.

Read OLI LOS Model
Read OLI geometric/LOS model.

Read LOS Geometric Grid
Read OLI LOS resampling grid.

7.2.10.7.2 Stage 2 - Setup Least Squares Matrices and Solve

For each input SCA, every residual for each input band combination that is not an outlier is mapped back to input space. These input space mappings are single value adjustments needed for each point to align the LOS, associated with the focal plane, between the bands of the combination. This mapping procedure is described in more detail below. Once all of these residuals are mapped back to the focal plane and stored within the least-squares (LSQ) matrices new LOSs can be calculated. PAN band residuals must be scaled by a factor of 2 to account for the resolution differences between the PAN band and the multispectral bands and the fact that the PAN residuals were measured in an image that had been resolution reduced to match the multispectral bands.

The matrices defining calibration the process takes the following form:

$$[A][coeff] = [Y]$$

The matrices [*A*] and [*Y*] shown above correspond to one tie point measurement. The matrix [*coeff*] are the unknown adjustments to the Legendre LOS coefficients, the matrix [*A*] contain the Legendre coefficient multipliers for the band combination corresponding to that one measurement, and the [*Y*] matrix contains the input space residuals for that one measurement. For one measurement the matrices have the following dimensions:
[coeff] = (2 * Number of Legendre * Number of bands) x 1 = M x 1
[A] = 2 x (2 * Number of Legendre * Number of bands) = 2 x M
[Y] = 2 x 1

$$[coeff] = \begin{bmatrix} a_{b1,0} \\ a_{b1,1} \\ a_{b1,2} \\ b_{b1,0} \\ b_{b1,1} \\ b_{b1,2} \\ a_{b2,0} \\ a_{b2,1} \\ a_{b2,2} \\ b_{b2,0} \\ \vdots \\ b_{b9,0} \\ b_{b9,1} \\ b_{b9,2} \end{bmatrix}$$

Where:
$a_{bi,j}$ = Legendre coefficient j for line direction (along track) for band i
$b_{bi,j}$ = Legendre coefficient j for sample direction (across track) for band i
j = 0, 1, 2 or the Number of Legendre coefficients to solve.
i = 1,2,...,9 (Number of OLI bands)

A 2x1 matrix pertaining to one residual measurement can be defined as:

$$[Y] = \begin{bmatrix} \Delta line \\ \Delta sample \end{bmatrix}$$

Where:
Δ*line* = input space residual in line direction (angular)
Δ*sample* = input space residual in sample direction (angular)

The input space residuals are calculated by finding the nominal (search) LOS in input space and the measured (search + measured offset) LOS in input space. These LOSs are found by mapping the output space line and sample locations to input space line and sample locations using the LOS geometric resampling grid (See OLI Resampling ADD) and then using the LOS model (see Model

Line-of-Sight Creation ADD) to convert the input space locations to LOSs. These input space nominal and measured locations are also used to construct the Legendre coefficient multipliers.

The design matrix [*A*] for one residual measurement is then:

$$[A_{nik}][coeff] = [Y_n]$$

$$[\mathbf{A}] = \begin{bmatrix} 0 \cdots 0 & -rl_{n,i,0} & \ldots & -rl_{n,i,j} & 0 & 0 & 0 & 0 \cdots 0 & sl_{n,k,0} & \ldots & sl_{n,k,j} & 0 & 0 & 0 & 0 \cdots 0 \\ 0 \cdots \cdots 0 & 0 & 0 & -rl_{n,i,0} & \ldots & -rl_{n,i,j} & 0 \cdots 0 & 0 & 0 & 0 & sl_{n,k,0} & \ldots & sl_{n,k,j} & 0 \cdots 0 \end{bmatrix}$$

Where:
$rl_{n,i,j}$ = reference band i Legendre polynomial
$sl_{n,k,j}$ = search band k Legendre polynomial
j = 0, 1, 2 or the Number of Legendre coefficients to solve
n = tie-point number

These matrices define one observation. A sequence of observations can be summed to define the normal equations for a set of coefficients that can be used to update the OLI LOS Legendre coefficients:

$$[N] = \sum A_{nik}^T W_{ik}^{-1} A_{nik}$$
$$[L] = \sum A_{nik}^T W_{ik}^{-1} Y_{nik}$$

Where [*N*] and [*L*] are summed over all *n* for all *i, k* band combinations. *W* is a weight matrix that is currently set to the same weight for all observations.

Since all of the tie point observations involve band differences, the solution lacks an absolute reference. To stabilize the solution a constraint observation is added to provide such a reference. This additional observation is required for the PAN band and represents an offset of zero for each direction (line and sample) of the PAN band. This allows the PAN band to be used as a reference and all other bands are then registered to it.

$$[A_{00}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \ldots & 0 & 0 & 0 \end{bmatrix}$$

$$[Y_{00}] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Where the PAN band is stored in the first two columns of the [A] observation matrix.

The bands that are not to be used in the solution process are removed by setting the corresponding diagonal elements of normal matrix [N] to the minimum number of points (given as an input value). The solution for a new set of Legendre coefficients is then:

$$[coeff] = [N]^{-1}[L]$$

Band Calibration Processing Steps

Note: Array indexes are zero-relative.

nLeg = Number of Legendre update coefficients to solve (1, 2, 3 valid options).
Matrix indexes are zero relative

Solve for New Focal Plane Parameters

Loop on each valid tie-point for each SCA and each band combination. Calculate LOS errors to determine per tie-point adjustment needed to LOS. Assimilate normal matrices and solve for updates needed to Legendre LOS, calculate new Legendre LOS based on updates from least-squares-solution. Calculate post and pre-fit statistics.

Calculate Line of Sight Angular Errors

Calculate delta input line and sample LOS needed for LSQ. Read elevation for tie-point, map tie-point to input space, and find adjustments needed between search and reference LOS vectors.

1. Initialize parameters

$$[W] = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

Where $\sigma^2 = 16$

2. For each SCA to process

Initialize pre-fit statistics variables
pre-fit sum line = 0
pre-fit sum sample = 0
pre-fit sum line^2 = 0
pre-fit sum sample^2 = 0
Initialize LSQ matrices to zero
[N] = [0]
[L] = [0]

2.1 For each band combination present
rband = reference band
sband = search band
2.1.1 For each tie-point
2.1.2 Calculate reference line, sample location and search adjusted line, sample location.
rline = tie-point reference line location
rsamp = tie-point reference sample location
sline = tie-point search line location + line offset measured
ssamp = tie-point search sample location + sample offset measured
Note: sline, ssamp is the adjusted (or true) search location.

Note that rline, rsamp, sline, ssamp are output space pixel locations.

2.1.3 Set rband and sband to zero-relative

rband = rband - 1

sband = sband - 1

Map residuals to input space (focal plane space).

2.1.4 Find elevation for reference and sample locations

relev = elevation at rline,rsamp

selev = elevation at sline,ssamp

2.1.5 Map rline,rsamp and sline,ssamp to input space using 3d_ols2ils (See Note #2) and the search band OLI resampling grid.

(riline,risamp) = 3d_ols2ils(search_grid, relev, rline, rsamp)

(siline,sisamp) = 3d_ols2ils(search_grid, selev, sline, ssamp)

Where

riline, risamp is the input space location of reference tie-point location.

siline, sisamp is the input space location of adjusted search tie-point location.

search_grid is the OLI resampling grid for the search band.

Note: Search band grid is used for mapping both the adjusted search (siline,sisamp) and the reference locations.

2.1.6 Calculate Legendre normalized detector location

$$\text{rnorm} = \frac{2.0 * \text{risamp}}{\text{number detectors in SCA} - 1} - 1$$

$$\text{snorm} = \frac{2.0 * \text{sisamp}}{\text{number detctors in SCA} - 1} - 1$$

rnorm = normalized reference detector

snorm = normalized adjusted search detector

2.1.7 Calculate reference and search along and across track LOS.

$$nom\_sear\_x = coef\_x_{s,0} + coef\_x_{s,1} * rnorm + coef\_x_{s,2} * (1.5 * rnorm^2 - 0.5)$$

$$nom\_sear\_y = coef\_y_{s,0} + coef\_y_{s,1} * rnorm + coef\_y_{s,2} * (1.5 * rnorm^2 - 0.5)$$

$$sear\_x = coef\_x_{s,0} + coef\_x_{s,1} * snorm + coef\_x_{s,2} * (1.5 * snorm^2 - 0.5)$$

$$sear\_y = coef\_y_{s,0} + coef\_y_{s,1} * snorm + coef\_y_{s,2} * (1.5 * snorm^2 - 0.5)$$

Where

ref_x, ref_y = along and across track view angles

sear_x, sear_y = along and across track view angles

$coef\_x_{s,n}$ = search Legendre along track coefficients

$coef\_y_{s,n}$ = search Legendre across track coefficients

2.1.8 Determine LOS vectors

sear_z = 1.0

$$m = \sqrt{sear\_x^2 + sear\_y^2 + sear\_z^2}$$

$$sear\_x = \frac{sear\_x}{m}$$

$$sear\_y = \frac{sear\_y}{m}$$

$$sear\_z = \frac{sear\_z}{m}$$

nom_sear_z = 1.0

$$m = \sqrt{nom\_sear\_x^2 + nom\_sear\_y^2 + nom\_sear\_z^2}$$

$$nom\_sear\_x = \frac{nom\_sear\_x}{m}$$

$$nom\_sear\_y = \frac{nom\_sear\_y}{m}$$

$$nom\_sear\_z = \frac{nom\_sear\_z}{m}$$

2.1.9 Calculate the LOS errors

2.1.9.1 Determine effective line-of-sight instantaneous-field-of-view (IFOV)

2.1.9.1.1 Map input search pixel location, line and sample, to output space.
sline = search line location
ssamp = search sample location
elevation = elevation for location sline, ssamp

Calculate elevation planes bounding current elevation.

$$zplane = \frac{elevation}{grid\ z\ spacing} + grid\ zero\ plane$$

*elev0 = grid z spacing * (zplane – grid zero plane)*
elev1 = elev0 + grid z spacing

Calculate cell index, row and column, for search line and sample location and zplane.
row = sline / grid cell line spacing
column = ssamp / grid cell sample spacing
*cell index0 = nrows * ncols * zplane + row * ncols + column*

Where:
grid z spacing = elevation difference between two grid planes
ncols = number of grid cell columns
nrows = number of grid cell rows

Calculate output space line, sample location for input space search line, sample location and zplane.
$a_{0,1,2,3}$ = grid sample location forward mapping coefficients for cell index0
$b_{0,1,2,3}$ = grid line location forward mapping coefficients for cell index0

*lms = sline * ssamp*

*osamp0 = a_0 + a_1 * ssamp + a_2 * sline + a_3 *lms*
*oline0 = b_0 + b_1 * ssamp + b_2 * sline + b_3 * lms*

Calculate cell index, row and column, for search line and sample location and zplane +1.

*cell index1 = nrows * ncols * (zplane + 1.0) + row * ncols + column*

Calculate output space line, sample location for input space search line, sample location and zplane+1.

$a_{0,1,2,3}$ = grid sample location forward mapping coefficients for cell index1
$b_{0,1,2,3}$ = grid line location forward mapping coefficients for cell index1

*lms = sline * ssamp*
*osamp1 = a_0 + a_1 * ssamp + a_2 * sline + a_3 *lms*
*oline1 = b_0 + b_1 * ssamp + b_2 * sline + b_3 * lms*

Calculate output space line, sample location for input space search line, sample location, and elevation.

w0 = (elev1 – elevation) / (elev1 – elev2)
w1 = (elevation – elev0) / (elev1 – elev2)

*$osamp_n$ = osamp0 * w0 + osamp1 * w1*
*$oline_n$ = oline0 * w0 + oline1 * w1*

2.1.9.1.2 Map input location ssamp, sline+1.0 to output space $osamp_{n+1}$,$oline_{n+1}$ (repeat step 2.1.9.1.1 for input location ssamp,sline+1)

2.1.9.1.3 Determine change in output space between input locations (ssamp, sline) and (ssamp, sline+1.0)
dline = $oline_n$ – $oline_{n+1}$
dsamp = $osamp_n$ = $osamp_{n+1}$
$$\text{distance} = \sqrt{\text{dline} * \text{dline} + \text{dsamp} * \text{dsamp}}$$

2.1.9.1.4
If earth acquisition calculate LOS distance to target

Calculate output latitude and longitude for search line and sample (see Forward Model section of LOS Projection Ellipsoid & Terrain).

Calculate time for current search line and sample (see section a.1 in Forward Model section of LOS Projection Ellipsoid & Terrain).

Calculate satellite position for current search line and sample time (see section a.4 in Forward Model section of LOS Projection Ellipsoid & Terrain).

Calculate target vector (see section a.7 in Forward Model section of LOS Projection Ellipsoid & Terrain).

LOS x coordinate = target x coordinate - satellite x coordinate
LOS y coordinate = target y coordinate - satellite y coordinate
LOS z coordinate = target z coordinate - satellite z coordinate

length = sqrt(LOS x * LOS x + LOS y * LOS y + LOS z * LOS z)

$IFOV_{along}$ = (output pixel size * distance) / length

If lunar acquisition

$IFOV_{along}$ = output pixel size * distance

2.1.9.1.5 Calculate delta input in radians

$$\Delta line = \frac{sear\_x}{sear\_z} - \frac{nom\_sear\_x}{nom\_sear\_z} + (siline - riline) * IFOV_{along}$$

$$\Delta samp = \frac{sear\_y}{sear\_z} - \frac{nom\_sear\_y}{nom\_sear\_z}$$

2.1.10 Create matrices need to sum with [N] and [L].

2.1.10.1 Calculate Legendre generating polynomial coefficients for search and reference:

sl_0 = 1.0
if(*nLeg* >= 2) sl_1 = snorm
if(*nLeg* == 3) sl_2 = 1.5 * snorm^2 - 0.5
rl_0 = 1.0
if (*nLeg* >= 2) rl_1 = rnorm
if(*nLeg* == 3) rl_2 = 1.5 * rnorm^2 - 0.5

Note: If the number of Legendre coefficients in the solution is less than 3 the corresponding sl_n and rl_n will be omitted.

2.1.10.2 Initialize [A] to zero and then set [A] indexes to sl_n and rl_n.

A[0][Number Legendre * sband + n] = sl_n
A[1][Number Legendre * sband + n] = sl_n
A[0][Number Legendre * rband + n] = -rl_n
A[1][Number Legendre * rband + n] = -rl_n

Where: n = 0 … *nLeg* -1
[A] = 0 elsewhere

2.1.10.3 Set [Y] according to input space deltas measured and sum pre-fit statistics.

2.1.10.3.1 Store deltas in [Y]

Y[0][0] = Δ*line*
Y[1][0] = Δ*samp*

2.1.10.3.2 Sum statistics

pre-fit sum line = pre-fit sum line + Δ*line*
pre-fit sum sample = pre-fit sum sample + Δ*sample*
pre-fit sum line^2 = pre-fit sum line^2 + Δ*line*^{2}
pre-fit sum sample^2 = pre-fit sum sample^2 + Δ*sample*^{2}

2.1.10.4 Create matrices to add to normal matrices

$[A_{tie\text{-}point}] = [A]^T[W][A]$

$[Y_{tie\text{-}point}] = [A]^T[W][Y]$

2.1.10.5 Sum N and L matrices

$[N] = [N] + [A_{tie\text{-}point}]$

$[L] = [L] + [Y_{tie\text{-}point}]$

2.2 Set minimum points for bands to omit from processing.

Eliminate observations for omitted band:

oband = band to omit - 1 (from earlier, bands are 1-relative)

$[N]_{g+n,i} = 0$

Where g = *nLeg* * 2 * oband

n = 0 … 2**nLeg* - 1

i = 0 … *nLeg* * 2 * Number of Bands - 1

$[N]_{i,g+n} = 0$

Where g = *nLeg* * 2 * oband

n = 0 … 2**nLeg* - 1

i = 0 … *nLeg* * 2 * Number of Bands - 1

$[N]_{g+n,g+n}$ = Minimum Points

Where g = *nLeg* * 2 * oband

n = 0 … 2**nLeg* - 1

$[L]_{g+n} = 0$

Where g = *nLeg* * 2 * oband

n = 0 … 2**nLeg* - 1

2.3 Solve for delta Legendre coefficients

$$[\Delta coeff] = [N]^{-1}[L]$$

2.4 Calculate new Legendre coefficients

$$\text{new along coeffs}_{\text{SCA,band,n}} = \text{previous along coeffs}_{\text{SCA,band,n}} + \sum_{i=0}^{(nLeg*NBAND)} \sum_{j=0}^{nLeg} [\Delta coeff_{i+j}]$$

$$\text{new across coeffs}_{\text{SCA,band,n}} = \text{previous across coeffs}_{\text{SCA,band,n}} = \sum_{i=nLeg}^{(nLeg*NBAND)} \sum_{j=0}^{nLeg} [\Delta coeff_{i+j}]$$

i = 0, *nLeg* , 2 * *nLeg*,…, NBANDS * *nLeg*

n, j = 0 … *nLeg* - 1

7.2.10.7.3 Stage 3 - Calculate Pre and Post fit Residuals

Compute Post-Fit Statistics

Calculate pre and post-fit statistics.

1. For each SCA calculate residuals

   Initialize post-fit statistics variables

   post-fit sum line = 0

   post-fit sum sample = 0

   post-fit sumsq line = 0

   post-fit sumsq sample = 0

2. For each band combination

2.1 Perform steps 2.1.1 - 2.1.10 from stage 2.

2.2 Calculate adjusted reference and search line/sample locations

riline' = $\Delta coeff_{sca,rband,0}$

if(*nLeg* >=2) riline' = riline' + rnorm * $\Delta coeff_{sca,rband,1}$

if(*nLeg* = = 3) riline' = riline' + (1.5 * rnorm^2 - 0.5) * $\Delta coeff_{sca,rband,2}$

risamp' = $\Delta coeff_{sca,rband,0}$

if(*nLeg* >= 2) risamp' = risamp' + rnorm * $\Delta coeff_{sca,rband,1}$

if(*nLeg* = = 3) risamp' = risamp' + (1.5 * rnorm^2 - 0.5) * $\Delta coeff_{sca,rband,2}$

siline' = $\Delta coeff_{sca,rband,0}$

if(*nLeg* >= 2) siline' = siline' + snorm * $\Delta coeff_{sca,sband,1}$

if(*nLeg* = = 3) siline' = siline' + (1.5 * snorm^2 - 0.5) * $\Delta coeff_{sca,sband,2}$

sisamp' = $\Delta coeff_{sca,sband,0}$

if(*nLeg* >= 2) sisamp' = sisamp ' + snorm * $\Delta coeff_{sca,sband,1}$

if(*nLeg* = = 3) sisamp' = sisamp' + (1.5 * snorm^2 - 0.5) * $\Delta coeff_{sca,sband,2}$

Where:

SCA, band, 0, 1, 2 are the SCA, band number and coefficients for the updates to the Legendre polynomials. The Δ*coeff* added to the riline are the along track updates the Δ*coeff* add to the risamp are the across track updates.

rband = index to reference band coefficient

sband = index to search band coefficient

2.3 Calculate new post fit Δ*errors* by updating Δ*line* and Δ*sample* with Legendre updates

Δ*line*' = Δ*line* + riline' - siline'

Δ*samp*' = Δ*samp* + risamp' - sisamp'

Where:

Δ*line* and Δ*samp* are the same as those calculated in 2.1.10 from stage 2. See note #10.

2.4 Sum post-fit variables

post-fit sum line = post-fit sum line + Δ*line*'

post-fit sum sample = post-fit sum sample + Δ*sample*'

post-fit sumsq line = post-fit sumsq line + Δ*line*'^{2}

post-fit sumsq sample = post-fit sumsq sample + Δ*sample*'^{2}

3. Calculate post and pre fit statistics for both line and sample directions:

$$\text{mean} = \frac{\text{sum}}{\text{number of points}}$$

$$\text{scale} = \frac{1}{\text{number of points} * (\text{number of points} - 1)}$$

$$\text{standard deviation} = \sqrt{\frac{\text{sum squares}}{\text{number of points} - 1} - \text{sum} * \text{sum} * \text{scale}}$$

$$\text{rmse} = \sqrt{\frac{\text{sum squares}}{\text{number of points}}}$$

Where

sum = pre/post sum line or pre/post sum sample

sum squares = pre/post sumsq line or pre/post sumsq sample

number of points = number of points used in LSQ fit

4. Create Band-to-Band Calibration output report (See table #1).

4.1 Write report header information.

4.2. Write post and pre-fit statistics (per SCA) for line and sample direction.

4.3. Write individual tie-point statistics (if tie-point reporting flag = Yes).
5. If CPF group flag is set to yes write out ASCII file of CPF group with new Legendre coefficients.

7.2.10.8 Output files

The output report contains a standard header. This standard header is at the beginning of the file and contains the following:

1) Date and time file was created.
2) Spacecraft and instrument pertaining to measurements.
3) Pointing (roll) angle of spacecraft/instrument.
4) Acquisition type
5) Report type (band-to-band)
6) Work order ID of process (left blank if not applicable)
7) WRS path/row
8) Software version that produced report.
9) L0R image file name

The following items should be stored (trended) in the database with respect to the Band-to-Band Calibration algorithm:

All report header information:
- Date and time
- Spacecraft instrument source
- Work order ID
- WRS path/row
- Software version
- Off-nadir angle
- L0Rp file name
- Processing file name

The following processing parameters:
- L0R product ID
- Bands processed
- SCAs processed

The following report file information:
- Number of points used per SCA
- Computed Legendre along track coefficient updates
- Computed Legendre across track coefficient updates
- New Legendre along track coefficients (updates + existing)
- New Legendre across track coefficients (updates + existing)
- Post-fit mean, standard deviation, RMSE
- Pre-fit mean, standard deviation, RMSE

See note #11.

Field	Description	Trend
Date and time	Date (day of week, month, day of month, year) and time of file creation.	Yes

Spacecraft and instrument source	LDCM and OLI	Yes
Processing Center	EROS Data Center SVT	No
Work order ID	Work order ID associated with processing (blank if not applicable)	Yes
WRS path/row	WRS path and row (See note #4)	Yes
Software version	Software version used to create report	Yes
Off-nadir angle	Off-nadir pointing angle of processed image file (See note #5)	Yes
Acquisition Type	Earth viewing or Lunar	Yes
L0Rp image file	L0Rp image file name used to create L1T	Yes
Processed image file name	Name of L1T used to create report	Yes
Number of Legendre coefficients	Number of Legendre coefficients present	Yes
Heading for pre and post fit statistics	One line of ASCII text defining pre and post statistics	
For each SCA (along and across track directions)		
SCA number	SCA number associated with statistics	Yes
Pre fit statistics	Mean, RMSE, standard deviation, along and across track direction (in units of radians)	Yes
Post fit statistics	Mean, RMSE, standard deviation, along and across track direction (in units of radians)	Yes
For each SCA and band		
Along track solution	Legendre along track correction coefficients	Yes
Across track solution	Legendre across track correction coefficients	Yes
For each SCA and band		
Along track updates	Updated Legendre along track coefficients	Yes
Across track updates	Updated Legendre across track coefficients	Yes
For each tie-point of each SCA and Band to process	Output produced only if tie-point results flag is set to Yes.	
Point ID	Point identifier	No
SCA number	SCA number for band combination	No
Reference output line	Output tie-point location in line direction	No
Reference output sample	Output tie-point location in sample direction	No
Reference input line	Reference input tie-point location in line direction	No
Reference input sample	Reference input tie-point location in sample direction	No
Search input line	Search input tie-point location in line direction	No
Search input sample	Search input tie-point location in sample direction	No
Reference band	Reference band	No
Search Band	Search band	
Measured line offset	Output space offset in line direction (from Band Accuracy Assessment residuals file)	No
Measured sample offset	Output space offset in sample direction (from Band Accuracy Assessment residuals file)	No
Pre-fit line delta	Pre-fit input space line delta/offset ($\Delta line$)	No
Pre-fit sample delta	Pre-fit input space sample delta/offset ($\Delta sample$)	No
Post-fit line delta	Post-fit input space line delta/offset ($\Delta line'$)	No
Post-fit sample delta	Post-fit input space sample delta/offset ($\Delta sample'$)	No

Table 1. Band Calibration Report file

If the CPF group creation flag is set to yes an ASCII file containing the updated Legendre LOS should be generated. This file would contain the new Legendre LOS for each SCA for every band and would be formatted according to the CPF group that the Legendre LOS resides in (for geometric prototype code this is the LOS_LEGENDRE group). The file would also contain the file attributes CPF group with the effective dates for the LOS generated (See note #3). The SCAs and bands that were not updated should still be represented within the file; these values should be the same for post and pre calibration.

7.2.10.9 Maturity

1. Band-to-Band Calibration for OLI closely follows that of ALIAS.
2. Band calibration for the cirrus band, involving lunar acquisitions, will take several steps. This will only include several pre-processing steps needed before band calibration at which point there will be a need for only one band calibration routine for both Earth and lunar based acquisitions, with the exception of possibly changing some processing parameters. These pre-processing steps will be performed with the CalVal Toolkit.

7.2.10.10 Notes

Some additional background assumptions and notes include:

1. The band calibration results currently contains the L1T name, pre and post fit mean, root mean squared error, and standard deviation for the along and across track direction of each SCA, new Legendre LOS coefficients, and a new CPF Legendre LOS group parameters. The individual tie-point characteristic information and (pre and post-fit) residuals and should be added to the report file (see table #1).
2. See "Using the LOS geometric resampling grid to map an output pixel location to an input pixel location" in the OLI Resampling ADD for ols2ils functionality.
3. The table listed below contains the file attributes and LOS groups that should be populated with the corresponding OLI fields when the CPF group creation flag is set to yes. The CPF_Status, CPF_Name_Source, CPF_Description, and CPF_Version fields were inserted during ALIAS development by software development, these fields may or may not be present/needed for OLI processing.

Parameter Groups	Parameter Name	Data Type	Description			Prelaunch Source
GROUP: LOS_LEGENDRE	Along_LOS_Legendre_B***BB***_***NNN***_SCA***SS***	float32 array (3 values) for each band of each SCA	Legendre polynomial coefficients defining along track viewing angle of band number ***BB***, band name ***NNN*** and SCA ***SS*** given in radians Valid format: for each term: SN.NNNNESN, where S = "+" or "-", N = 0 to 9, and E = "E".			
GROUP: LOS_LEGENDRE	Across_LOS_Legendre_B***BB***_***NNN***_SCA***SS***	float32 array (3 values) for each band of each SCA	Legendre polynomial coefficients defining across track viewing angle of band number ***BB***, band name ***NNN*** and SCA ***SS*** given in radians Valid format: for each term: SN.NNNNESN, where S = "+" or "-", N = 0 to 9, and E = "E"			

The file name for the CPF group can follow the convention of:
Legendre_coefficients_<effective begin date>_<effective end date>.odl
Where:
effective begin date = YYYYMMDD
effective end date = YYYYMMDD
YYYY = Year
MM = Month of year
DD = Day of month

4. Any kind of "non-WRS" collect; lunar or off-nadir viewing at the poles should have 000/000 listed as the path/row.
5. Pointing angle for lunar acquisitions would be 0.0.
6. Currently it is not expected that any calibration will be done on anything other than the full range of Legendre coefficients (3), however the prototype code supports the range of 1-3 Legendre coefficients in the solution. The OLIAS prototype code will keep this option and it should remain in the system.

7.2.11 OLI Focal Plane Alignment Calibration

7.2.11.1 Background/Introduction

The OLI focal plane alignment calibration algorithm compares a precision and terrain corrected (L1T) OLI panchromatic band image of a geometric calibration site with the corresponding reference image. Reference images will be constructed (offline) from mosaics of high resolution digital orthophoto quad (DOQ) data sets or other higher (than OLI) resolution data sources (e.g., SPOT). Whatever the source, these high accuracy reference data sets will be collectively referred to as "DOQ" images. Each separated-SCA L1T image is compared to the reference to measure SCA-specific deviations from the scene-average registration. The measured deviations are used to estimate corrections to the Legendre polynomial coefficients that model the nominal panchromatic band lines-of-sight for each SCA.

The algorithm is implemented in two steps: 1) a mensuration/setup step in which the separated-SCA L1T image is correlated with the reference image to measure the within-SCA deviations, and; 2) a calibration Legendre coefficient update computation step in which the measured deviations are used to compute line-of-sight model correction Legendre coefficients that adjust the original LOS model to minimize the residual image deviations. The calibration update step includes applying an outlier filter to the image measurements. Separating the algorithm into two distinct steps makes it possible to run the calibration update step multiple times, using different outlier filter thresholds, for example, without having to perform the time consuming image mensuration/correlation setup procedure more than once.

Results from individual calibration scenes are stored in the geometric trending database so that results from multiple scenes can be analyzed together when deciding whether and how to adjust the operational focal plane calibration. When a focal plane calibration update is generated, the other spectral bands would subsequently be re-registered to the panchromatic band using the band alignment calibration procedure.

The OLI focal plane calibration procedure is derived from the ALI focal plane calibration algorithm used in ALIAS. The prototype LDCM implementation is very similar to the ALIAS focal_plane_setup, which measures the SCA-specific deviations relative to the reference image, and focal_plane_legendre, which calculates the Legendre polynomial coefficient updates, applications.

7.2.11.2 Dependencies

The OLI focal plane alignment calibration algorithm assumes that the L1T process flow has created a substantially cloud-free SCA-separated (nadir-viewing) path-oriented L1T panchromatic band image, over a geometric calibration site, which has been registered to a reference image by using DOQ control in the LOS model correction procedure. Note that either the L1T image will be generated to exactly match the reference DOQ image frame or the DOQ reference image will have to be resampled to match the L1T as a preprocessing step. This algorithm also assumes that the CPF, precision LOS model, precision grid file, and DEM used to produce the L1T image, are available.

7.2.11.3 Inputs

The OLI focal plane alignment calibration algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data). The second column shows which algorithm step (image mensuration or correction model computation) uses the input.

Algorithm Inputs	**Processing Step**
ODL File (implementation)	Both
Calibration Parameter File (CPF) Name	Both
L1T Image File Name	Step 1
Precision LOS Model File Name	Step 1
Precision LOS Grid File Name	Step 1
DEM File Name	Step 1
Reference Image File Name	Step 1
Correlation Data File Name	Both
Report File Name (see note #5)	Step 2
Number of Tie Points per Cell	Step 1
Outlier tolerance	Step 2
L0Rp ID (for trending)	Step 2
Work order ID (for trending)	Step 2
WRS Path (for trending)	Step 2
WRS Row (for trending)	Step 2
Calibration effective dates for updated parameters	Step 2
Trending flag	Step 2
CPF	Both
Algorithm Parameters (formerly system table parameters)	
Size of Correlation Window (see note #5)	Step 1
Peak Fit Method	Step 1
Min Correlation Strength (see note #5)	Step 1
Max Correlation Displacement (see note #5)	Step 1
Fill Threshold Fraction (max percent of window containing fill value)	Step 1
Tie point weight (in units of 1/microradians^2)	Step 2
Fit order (see note #5)	Step 2
Post-fit RMSE Thresholds (trending metrics)	Step 2
Precision Grid File (see LOS Projection ADD for details)	Step 1
Number of SCAs	Step 1
For each SCA:	Step 1
Grid cell size in lines/samples	Step 1

Number of lines/samples in grid	Step 1
Number of z-planes, zero z-plane index, z-plane spacing	Step 1
Array of grid input line/sample locations	Step 1
Array of output line/sample locations (per z-plane)	Step 1
Array of forward mapping coefficients	Step 1
Array of inverse mapping coefficients	Step 1
Rough mapping polynomial coefficients	Step 1
Precision LOS Model File (see LOS Model Creation ADD for details)	Step 1
OLI Pan Along-Track IFOV (in radians)	Step 1
Number of SCAs	Step 1
Number of Bands	Step 1
Number of Detectors per SCA per Band	Step 1
Focal Plane Model Parameters (Legendre Coefficients) (in radians)	Step 1
L1T Image (separated SCA)	Step 1
Image corner coordinates	Step 1
Pixel size (in meters)	Step 1
Image size	Step 1
Search image pixel data (panchromatic)	Step 1
DEM	Step 1
DEM corner coordinates	Step 1
Pixel size (in meters)	Step 1
DEM size	Step 1
Elevation data	Step 1
Reference Image	Step 1
Image corner coordinates	Step 1
Pixel size (in meters)	Step 1
Image size	Step 1
Reference image pixel data	Step 1
Correlation Data File (output of Step 1)	Step 2
Correlation results in input space pixels	Step 2
LOS errors in radians	Step 2
Correlation results in output space pixels	Step 2

7.2.11.4 Outputs

Step 1: Focal Plane Alignment Setup
Correlation Data File (temporary output passed to Legendre step)
Correlation results in output space pixels
Correlation results in input space pixels
LOS errors in radians
Step 2: Focal Plane Alignment Legendre (see Table 1 below)
Report File (see Table 1 below for details)
Standard report header
Acquisition date
Ref (DOQ)/Search (L1T) image names
Number of SCAs
For each SCA:
SCA Number
Old Along- and Across-track Legendre coefficients (NSCAx2x3)
Along- and Across-track Legendre error (fit) coefficients (NSCAx2x3)
New Along- and Across-track Legendre coefficients (NSCAx2x3)
Pre-fit along- and across-track residual statistics (mean, stddev, RMSE)

Post-fit along- and across-track residual statistics (mean, stddev, RMSE)
Confidence level used for outlier rejection
Legendre polynomial fit order
Number of tie points used for current SCA
CPF LOS_LEGENDRE Group (ODL format file)
Effective Dates (embedded in file name)
New Legendre polynomial coefficients (NSCAx2x3)
Measured Tie Point Data
For each point:
SCA Number
Grid Cell Column Number
Nominal Output Space Line
Nominal Output Space Sample
Measured LOS Error Delta Line (in pixels)
Measured LOS Error Delta Sample (in pixels)
Measured LOS Error Along-Track Delta Angle (in microradians)
Measured LOS Error Across-Track Delta Angle (in microradians)
Tie Point State (outlier) Flag
Along-Track Fit Residual (in microradians)
Across-Track Fit Residual (in microradians)
Focal Plane Alignment Trending Database (see Table 1 below for details)
L0Rp ID
Work Order ID
WRS path/row
Acquisition date
Ref (DOQ) image name
Number of SCAs
For each SCA:
SCA Number
Old Along- and Across-track Legendre coefficients (NSCAx2x3)
Along- and Across-track Legendre error (fit) coefficients (NSCAx2x3)
New Along- and Across-track Legendre coefficients (NSCAx2x3)
Pre-fit along- and across-track residual statistics (mean, stddev, RMSE)
Post-fit along- and across-track residual statistics (mean, stddev, RMSE)
Confidence level used for outlier rejection
Number of tie points used for current SCA

7.2.11.5 Options

Focal Plane Alignment Calibration Trending On/Off Switch

7.2.11.6 Prototype Code

Input to the executable is an ODL file; outputs are a binary tie point mensuration file (used internally only), an ASCII report file, an ASCII ODL-formatted CPF fragment, and trending data written to the stdout and captured in an ASCII log file.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The code units of the prototype implementation are briefly described here. Additional details are provided below for units that perform core algorithm processing logic.

Focal_Plane_Setup.c - This routine is the main driver for the setup portion of the focal plane calibration.

get_focal_plane_parms.c - This function gets the input parameters from the ODL parameter file. It is used by both the setup and legendre executables.

check_images_match.c - This function checks to make sure the L1T search, DOQ reference, and DEM images all match. The corners of all the images should match to within half a pixel, and the reference and L1T search image should be the same resolution (pixel size) and the same size. This function is an initial check to make sure that all the images are consistent before correlation is attempted. This function assumes the DOQ reference image and the DEM are one-band images.

set_up_grid.c - This function reads the grid file into the grid data structure. The whole grid structure is returned so the caller can free all memory allocated when the grid was read using the grid deallocation call (oli_free_grid).

select_corr_pts.c - This function selects nominal correlation points evenly distributed about the center point of each grid cell (output space).

calc_input_space_errors.c - This function calculates the errors in input space pixels. This is done by first correlating in output space and converting the correlated locations to input space.

perform_correlation.c - This function performs the normalized gray-scale correlation at each point. It does this by invoking the correlation library routines described in the GCP Correlation Algorithm Description Document (e.g., math_submit_chip_to_corr).

map_coords_to_input_space.c - This function uses the inverse mapping coefficients in the grid to calculate the input space line/sample for each output space line/sample.

calc_los_errors.c - This function uses the tie point reference and search input space locations to calculate the angular line-of-sight errors.

output_correlation_info.c - This function writes the tie point correlation results to a file.

Generate_Legendre_Polynomials.c - This routine is the main driver for the Legendre polynomial generation portion of focal plane calibration.

read_correlation_info.c - This function reads the correlation information from the file generated by the Output Correlation Information sub-algorithm above.

filter_outliers.c - This function separates the focal plane correlation data into groups for each SCA for the X (sample) and Y (line) directions. It then finds the standard deviation for the points in each group. Outlier rejection is then performed on the points based on the tolerance selected by the user and the Student's T distribution. This procedure is described in the Geometric Accuracy Assessment Algorithm Description Document.

calculate_point_weights.c - This function calculates the weight associated with each correlation point for doing the Legendre polynomial fit. Currently, this routine assigns the weight passed in to each point, effectively assigning each point an equal weight. Originally, it was thought that the correlation strength would factor into the weight, but that was determined to not be needed. This routine was left in to allow point-specific weight factors to be added at a later date.

fit_polynomials.c - This function performs the weighted least squares fit of the correlation data points (using the angular error) to find the Legendre error polynomial.

calculate_post_fit_residuals.c - This function calculates the residual statistics after the Legendre polynomial coefficients have been fit. This unit includes the calc_legendre_poly() function that calculates the Legendre polynomial for the input normalized detector value.

create_focal_plane_report.c - This function generates a file reporting the results of the SCA Legendre polynomial fit calculations. The report file contents are shown in Table1 below. This unit also includes the write_coeffs() function that writes an entire set of coefficients to the indicated output file.

trending_dummy.c - This function is a placeholder for the logic that will write the results of the SCA Legendre polynomial fit calculations to the geometric characterization database. In the prototype implementation the actual database output is replaced by dummy ASCII output to stdout.

write_SCA_parameters_cpf.c - This function writes the updated LOS_LEGENDRE parameter group of the CPF, in the ODL format used by the CPF, to an output file.

7.2.11.7 Procedure

The Focal Plane Alignment Calibration Algorithm is used for on-orbit calibration of the alignment of the lines-of-sight of the SCAs relative to each other. This calibration is necessary to meet the image registration, geodetic accuracy, and geometric accuracy requirements.

Procedure Overview
The focal plane alignment algorithm adjusts each SCA to a known stable reference. By aligning each SCA to a common reference, any measured inter-SCA misalignment is removed. Each SCA is correlated against a reference image created from a mosaic of digital orthophoto quadrangle (DOQ) images. A new set of 2^{nd} order Legendre LOS coefficients, representing updates or corrections to the original polynomials, are generated by fitting a set of coefficients to the residuals.

Substantially cloud free scenes should be used for focal plane alignment calibration. The imagery should have ground control applied and terrain displacements removed, i.e. the imagery should be a terrain corrected (L1T) data set.

Stage 1: Setup – Correlate L1T Image with DOQ Reference
An array of test points is generated for each SCA based upon the number of points per grid cell specified in the input parameters. The OLI geometric grid is used to generate the test point array by spacing the test points at regular intervals in input space, and then computing the corresponding output space coordinates for each. Constructing the test point array in input space ensures that the test points fall within the active area of each SCA.

Image windows extracted from the L1T image at the test point locations are correlated with corresponding windows extracted from the reference DOQ image, using normalized gray scale

correlation. This procedure is the same as that described in the OLI Image Registration Accuracy Assessment Algorithm Description Document. Since the expected offsets are small, the L1T and DOQ image windows are the same size. The correlation procedure yields measured deviations (or correlation failure flag) in the line and sample directions, estimated to sub-pixel accuracy. These deviations, or residuals, are in units of output space pixels.

The residuals measured in output space are converted to differences in LOS along- and across-track angles by mapping the reference point location from output space to input space and then mapping the search point location from output space to input space. The mappings are performed using the OLI geometric grid that was used to resample the L1T image, and include the test point elevation interpolated from the input DEM. This three-dimensional output space to input space mapping (3d_ols2ils) is described in the OLI Image Resampling Algorithm Description Document. Once an input space location is found for both points, the LOS vectors are calculated for each input sample location using the OLI LOS model. This is described in the Find LOS section of the OLI LOS Projection Algorithm Description Document.

The angular differences in input space are then:

$$\text{along track residual} = \frac{r_x}{r_z} - \frac{s_x}{s_z} + (\text{input reference line} - \text{input search line}) * \text{along track IFOV}$$

$$\text{across track residual} = \frac{r_y}{r_z} - \frac{s_y}{s_z}$$

where:

r_x, r_y, r_z = reference x,y,z vector components of LOS

s_x, s_y, s_z = search x,y,z vector components of LOS

input reference line = input line location for reference point

input search line = input line location for search point

Stage 2: Legendre – Compute Focal Plane Calibration Update

A weighted least squares routine is used to generate the fit between the angular residuals and the updated Legendre polynomial coefficients. The weight matrix [**W**] is an NxN diagonal matrix where the diagonal elements are a user entered weight value.

$$\underset{NxN}{[W]} = \begin{bmatrix} w & 0 & \cdots & 0 & 0 \\ 0 & w & 0 & \cdots & 0 \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ 0 & \cdots & \cdots & w & 0 \\ 0 & 0 & \cdots & 0 & w \end{bmatrix}$$

where w = user entered weight

The weight matrix was included in order to make it possible to differentially weight the measured deviations based on correlation strength, but this is not implemented in the baseline algorithm. Instead, a common weight, read from the CPF, is used for all points.

The observation matrix is an Nx1 matrix containing the correlation residuals in input space angular units. There are two observation matrices, one for the along track residuals and one for the across track residuals.

$$\underset{Nx1}{[Y]} = \begin{bmatrix} \text{residual}_0 \\ \text{residual}_1 \\ \vdots \\ \text{residual}_N \end{bmatrix}$$

The design matrix is an Nx3 matrix with each row of the matrix being equal to the Legendre polynomial term associated with the reference sample location of the corresponding residual measurement. The calculation of these Legendre polynomial terms, as functions of the input sample location, is described below.

$$[X] = \begin{bmatrix} l_{0,0} & l_{1,0} & l_{2,0} \\ l_{0,1} & l_{1,1} & l_{2,1} \\ l_{0,2} & l_{1,2} & l_{2,2} \\ \vdots & \vdots & \vdots \\ l_{0,N} & l_{1,N} & l_{2,N} \end{bmatrix}$$

where:

$l_{i,j}$ = i^{th} Legendre polynomial term associated with reference sample location of the j^{th} residual measurement.

The solution for the updates to the original Legendre LOS coefficients can be found from:

$$[coeff] = \left([X]^T [W][X]\right)^{-1}\left([X]^T [W][Y]\right)$$

The matrix [*coeff*] is a 1x3 matrix containing the corrections to be applied to the original Legendre coefficients. A separate solution is found for the along and across track component. The mean offset across all the SCAs corresponds to the mean residual precision correction error for the calibration scene. It is subtracted from the coefficients generated for each SCA to avoid introducing any net pointing bias into the focal plane calibration. Once the mean is subtracted, the corrections are then added to the original Legendre LOS coefficients to compute the updated focal plane alignment parameters.

Only the PAN band is used for focal plane alignment. Band calibration uses the PAN band as the reference for all the other bands, thus the multispectral bands are aligned to the PAN band during band calibration. A band alignment calibration should be performed following an update to the focal plane (pan band) calibration to avoid degrading the band-to-band registration.

Figure 1 shows the architecture for the setup portion of the Focal Plane Alignment Calibration algorithm.

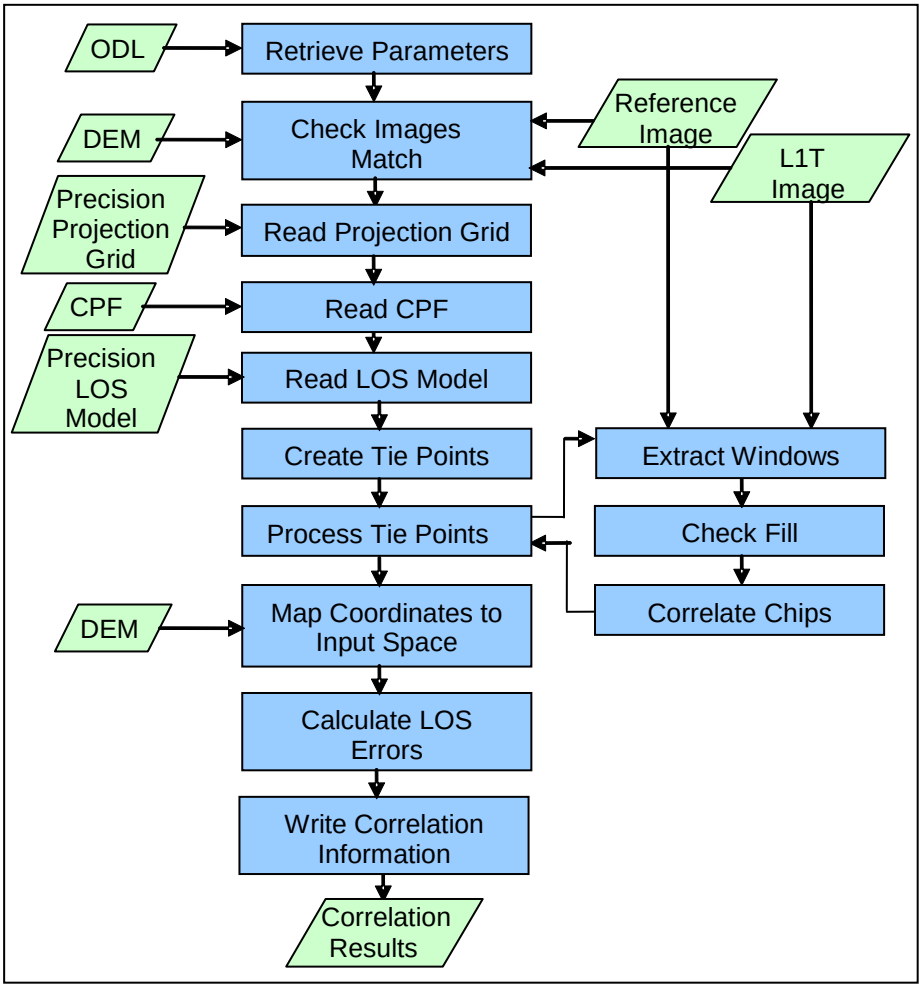


Figure 1: Focal Plane Calibration Setup Algorithm Architecture

Figure 2 shows the architecture of the Legendre polynomial fitting portion of the Focal Plane Alignment Calibration algorithm.

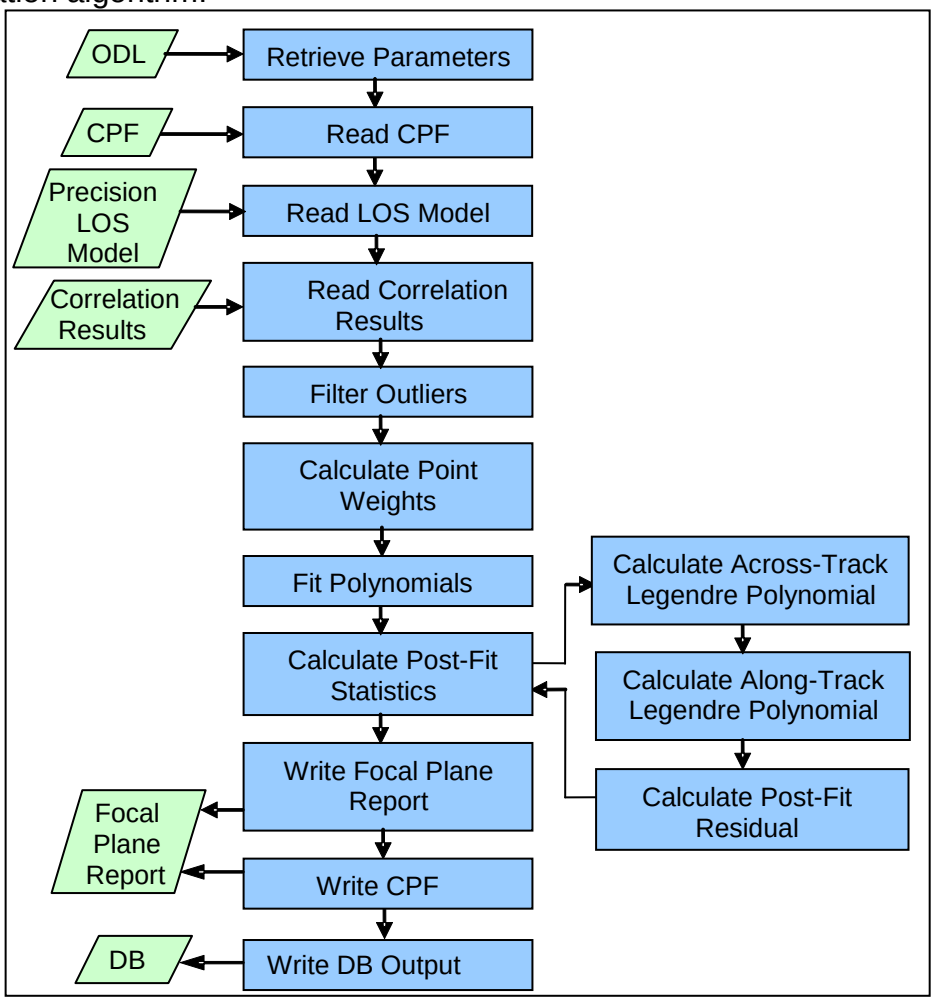


Figure 2: Focal Plane Calibration Legendre Polynomial Generation Algorithm Architecture

Focal Plane Cal Setup Sub-Algorithm

The setup portion consists of correlating points between the search image and the reference image, and converting the correlation offsets into line-of-sight deviations that can be used to model each SCA's detector array, modeled by a quadratic Legendre polynomial. The results of this program are used in the second portion of focal plane calibration, the generation of new Legendre polynomials. This program creates a temporary output file that is read by the second portion of focal plane calibration.

Select Correlation Points Sub-Algorithm

To ensure evenly distributed tie point locations during correlation, locations are defined to lie at the center of each resampling grid cell, or sub-cell. There will be pts_per_cell equally-spaced points per grid cell. For example, if there are 4 points per cell, they will be placed as shown in Figure 3.

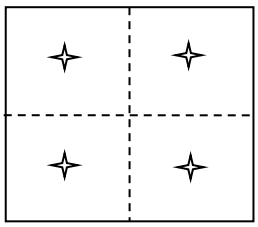


Figure 3: Correlation Point Placement in Grid Cell

The cell is divided into a 2-by-2 grid of 4 sub-cells, and each sub-cell is divided in half to place the point in the middle, yielding points at (0.25,0.25), (0.75,0.25), (0.25,0.75), and (0.75,0.75).

The calculation of the output space line/sample coordinates of the tie points is done as follows:

a) Compute number of rows and columns of tie points in each cell.
ncol = (int)ceiling(sqrt(pts_per_cell))
nrow = (int)ceiling((double)pts_per_cell/(double)ncol)
This creates an array of tie points containing at least pts_per_cell points.

b) For each tie point, i = 1 to ncol and j = 1 to nrow:

b1) Compute the grid cell fractional location (cfrac,rfrac).

$$cfrac = \frac{2i-1}{2\ ncol} \qquad rfrac = \frac{2j-1}{2\ nrow}$$

b2) Compute the output space line, ol_{ij}, using bilinear interpolation on the output line numbers at the grid cell corners, where l_{UL}, l_{UR}, l_{LL}, and l_{LR} are the output space line coordinates at the grid cell upper-left, upper-right, lower-left, and lower-right corners, respectively:

$ol_{ij} = l_{UL} * (1\text{-cfrac})*(1\text{-rfrac})$
$+ l_{UR} * cfrac * (1\text{-rfrac})$
$+ l_{LL} * (1\text{-cfrac}) * rfrac$
$+ l_{LR} * cfrac * rfrac$

b3) Compute the output space sample, os_{ij}, using bilinear interpolation on the output sample numbers at the grid cell corners, where s_{UL}, s_{UR}, s_{LL}, and s_{LR} are the output space sample coordinates at the grid cell upper-left, upper-right, lower-left, and lower-right corners, respectively:

$os_{ij} = s_{UL} * (1\text{-cfrac})*(1\text{-rfrac})$
$+ s_{UR} * cfrac * (1\text{-rfrac})$
$+ s_{LL} * (1\text{-cfrac}) * rfrac$
$+ s_{LR} * cfrac * rfrac$

Note that the bilinear weights are the same for the line and sample computations and only need be computed once.

The heritage version of this sub-algorithm locates the points by computing the intersection of the cell diagonals and then calculating offsets from that point. It is more straightforward to simply use bilinear interpolation, as described above, to calculate the tie point output space coordinates, so this unit has been reworked from the heritage implementation (see note #2).

Map Coordinates to Input Space Sub-Algorithm

This function uses the inverse mapping coefficients in the grid to calculate the input space line/sample for each output space line/sample. It does this for both the reference image line/sample location and the search (L1T) image line/sample location.

For each SCA

For each residual

1) Interpolate a height from the DEM at the location corresponding to the tie point reference image line/sample coordinates (xxx_get_elevation).

2) Map the reference output line/sample location to its corresponding input line/sample location using oli_3d_ols2ils routine

3) Interpolate a height from the DEM at the location corresponding to the tie point search image line/sample coordinates (xxx_get_elevation).

4) Map the search output line/sample location to its corresponding input line/sample location using the oli_3d_ols2ils

Calculate LOS Errors Sub-Algorithm (calc_los_errors)

This function uses the tie point reference and search input space locations to calculate the angular line-of-sight errors.

For each SCA

For each residual

1) Calculate reference line of sight vector for sample location using the nominal detector type and the precision LOS model (oli_findlos).

2) Calculate the search LOS vector for sample location using the nominal detector type and the precision LOS model.

3) Calculate the residual errors in terms of the difference in the LOS along and across track angles:

along-track LOS error = ref los.x/los.z – srch los.x/los.z
+ input line error * along-track pan IFOV

across-track LOS error = ref los.y/los.z – srch los.y/los.z

Output Correlation Information Sub-Algorithm

The correlation points are dumped to a binary file, so the second phase of focal plane calibration (Legendre polynomial generation) can read them directly back in. First, a long integer is written to indicate the number of records, then all the records are written. Each record contains the following fields:

Type	Field	Description
Int	sca_number	SCA number (0-relative)
Int	grid_column	grid column number (0-relative)
Int	grid_row	grid row number (0-relative)

double	nom_os_pt.line	nominal output space point line
double	nom_os_pt.samp	nominal output space point sample
double	ref_os_pt.line	reference output space line
double	ref_os_pt.samp	reference output space sample
double	srch_os_pt.line	search output space line
double	srch_os_pt.samp	search output space sample
double	ref_is_pt.line	reference input space line
double	ref_is_pt.samp	reference input space sample
double	srch_is_pt.line	search input space line
double	srch_is_pt.samp	search input space sample
double	los_err.line	angular along-track LOS error
double	los_err.samp	angular across-track LOS error
double	los_err_pix.line	line LOS error in pixels
double	los_err_pix.samp	sample LOS error in pixels
double	correlation_accuracy	correlation accuracy
ActiveFlag	active_flag	correlation success flag
double	pt_weight	point weight for use in fit
double	fit_residual.line	line residual from fit of pts
double	fit_residual.samp	sample residual from fit of pts

This is not a human-readable (ASCII) file, because it is only used to transport information from the first phase of calibration to the second. If the file already exists, it will be overwritten.

Generate Legendre Polynomials Sub-Algorithm
The generate Legendre portion reads the results of the focal plane setup, filters the outliers, fits the data to a Legendre polynomial, updates the SCA models, and generates output reports. This process is outlined below.

a) For each SCA

a.1) Build design matrix

Calculate normalized detector for reference sample location (see note #1). Note that in this context the “detector” number is the input sample number within the current SCA.

$$\text{normalized detector} = \frac{2 * \text{detector}}{\text{number of detectors}} - 1$$

where:
detector = reference sample location
number of detectors = number of pan band detectors in current SCA

Calculate “row” of design matrix and store in matrix

$l_{0,j} = 1$
$l_{1,j}$ = normalized detector
$l_{2,j} = 1.5 * (\text{normalized detector})^2 - 0.5$

where j = residual number

a.2) Build weight matrix using the (fixed) input weight value.

a.3) Build line and sample observation matrices from residuals in terms of angular differences.

a.4) Solve for line and sample solutions using weighted least squares routine (see the Fit Polynomials sub-algorithm below).

a.5) Calculate pre-fit residual statistics from the original measured deviations.

a.5.1) Calculate statistics for along- and across-track residuals used in a.3.

Compute mean, standard deviation and RMSE for the residuals. Values are calculated for along- and across-track directions independently.

For along-track residuals:

a.5.1.1) Calculate mean
a.5.1.2) Calculate standard deviation
a.5.1.3) Calculate RMSE

For across-track residuals:

a.5.1.4) Calculate mean
a.5.1.5) Calculate standard deviation
a.5.1.6) Calculate RMSE

a.6) Calculate post fit residuals statistics for correction coefficients

Post-fit residuals are calculated by updating the original residual deviations used in step a.3 above using the Legendre polynomial corrections. Statistics are then calculated on these updated residuals.

a.6.1) For each residual

a.6.1.1) Calculate normalized detector for reference sample location (as shown in a.1)

a.6.1.2) Calculate updated along-track LOS angle from along-track Legendre coefficients calculated in a.4 (see Calculate Legendre Polynomial sub-algorithm below).

a.6.1.3) Find difference between along-track angle from a.6.1.2 and along-track angular residual

a.6.1.4) Calculate updated across-track LOS from across-track Legendre coefficients calculated in a.4

a.6.1.5) Find difference between across-track angle from a.6.1.4 and across-track angular residual

a.6.2) Calculate statistics for along and across track updated residuals calculated in a.6.1.

Compute mean, standard deviation and RMSE for updated residuals. Values are calculated for along- and across-track directions independently.

For along-track residuals:

a.6.2.1) Calculate mean
a.6.2.2) Calculate standard deviation
a.6.2.3) Calculate RMSE

For across-track residuals:

a.6.2.4) Calculate mean
a.6.2.5) Calculate standard deviation
a.6.2.6) Calculate RMSE

b) Remove bias from the correction coefficients

sum along = sum across = 0.0

b.1) For all SCAs:

sum along = sum along + $coeff\_along_{0,sca}$

sum across = sum across + $coeff\_across_{0,sca}$

b.2) Compute the average across all SCAs:

$$\text{mean along} = \frac{\text{sum along}}{\text{number of scas}}$$

$$\text{mean across} = \frac{\text{sum across}}{\text{number of scas}}$$

b.3) For all SCAs:

$coeff\_along_{0,sca}$ = $coeff\_along_{0,sca}$ – mean along

$coeff\_across_{0,sca}$ = $coeff\_across_{0,sca}$ – mean across

c) For each SCA, add the correction coefficients to original Legendre LOS coefficients:

new along $legendre_{sca,i}$ = update along $legendre_{sca,i}$ + old along $legendre_{sca,i}$

new across $legendre_{sca,i}$ = update across $legendre_{sca,i}$ + old across $legendre_{sca,I}$

where:
i = 0,1,2 Legendre polynomial number
sca = SCA number

Filter Outliers Sub-Algorithm

This function separates the focal plane correlation data into groups for each SCA for the X (sample) and Y (line) directions. It then finds the standard deviation for the points in each group. Outlier rejection is then performed on the points based on the tolerance selected by the user and the Student's T distribution. This procedure is described in the Geometric Accuracy Assessment Algorithm Description Document.

Calculate Point Weights Sub-Algorithm

This function calculates the weight associated with each correlation point for doing the Legendre polynomial fit.

Currently, this routine assigns the weight passed in to each point, effectively assigning each point an equal weight. Originally, it was thought that the correlation strength would factor into the weight, but that was determined to not be needed. This routine was left in to allow point-specific weight factors to be added at a later date.

Fit Polynomials Sub-Algorithm

This function performs the weighted least squares fit of the correlation data points (using the angular error) to find the Legendre error polynomial. The least squares correction parameter vector **X** is given by solving:

$$\mathbf{A}^T\mathbf{WAX} = \mathbf{A}^T\mathbf{WY}$$

Where:

A is the [corr_points x NUMBER_COEFFS] design matrix

W is the weight matrix including the weight for each correlation data point of the **A** matrix (only the diagonal contains weights for that row)

$\mathbf{A}^T$ being the transpose of the **A** matrix

X is the [NUMBER_COEFFS x 1] matrix with the Legendre coefficients we are looking for

Y is the [corr_points x 1] matrix of angular errors for each correlation data point corresponding to the rows in the **A** matrix

Solving the above equation for **X** yields: $\mathbf{X} = (\mathbf{A}^T\mathbf{WA})^{-1} * \mathbf{A}^T\mathbf{WY}$

An optimization is used to keep the size of the matrices to a manageable level. Since the [**W**] matrix (weight) only contains data on the diagonal the $\mathbf{A}^T\mathbf{WA}$ normal equation matrix can be accumulated in a [NUMBER_COEFFS x NUMBER_COEFFS] matrix and the $\mathbf{A}^T\mathbf{WY}$ matrix can be accumulated in a [NUMBER_COEFFS x 1] matrix. The $\mathbf{A}^T\mathbf{W}$ and $\mathbf{A}^T\mathbf{WA}$ matrices are the same for both along- and across-track directions.

Calculate Legendre Polynomial Sub-Algorithm

This function calculates the Legendre polynomial for the input normalized detector value, x:

$along = coeff\_along_0 + coeff\_along_1\, x + coeff\_along_2\, (1.5*x^2 - 0.5)$

$across = coeff\_across_0 + coeff\_across_1\, x + coeff\_across_2\, (1.5*x^2 - 0.5)$

Write Focal Plane Calibration Results to Characterization Database
This function writes the results of the SCA Legendre polynomial fit calculations to the geometric characterization database. The output is only written to the database if the post-fit along- and across-track RMSE statistics are all below the threshold values specified in the CPF (the trending metrics). The characterization database output is listed in Table 1 below.

Write SCA Parameters CPF Sub-Algorithm
This function writes the updated LOS_LEGENDRE parameter group of the CPF, in the ODL format used by the CPF, to a separate output file named “LOS_LEGENDRE_yyyymmdd_YYYYMMDD.cpf”, where yyyymmdd is the user-specified start effective date and YYYYMMDD is the user-specified ending effective date. Current plans call for actual calibration updates to be based on multiple scene results extracted from the characterization database, so this capability is primarily a convenience for testing purposes.

Algorithm Output Details
The contents of the output focal plane alignment calibration report file and the corresponding geometric characterization database outputs are summarized in Table 1 below. All fields are written to the output report file but only those with "Yes" in the "Database Output" column are written to the characterization database. Note that the first eleven fields listed constitute the standard report header.

Field	Description	Database Output
Date and time	Date (day of week, month, day of month, year) and time of file creation.	Yes
Spacecraft and instrument source	LDCM and OLI	Yes
Processing Center	EROS Data Center SVT (see note #4)	Yes
Work order ID	Work order ID associated with processing (blank if not applicable)	Yes
WRS path	WRS path number	Yes
WRS row	WRS row number	Yes
Software version	Software version used to create report	Yes
Off-nadir angle	Scene off-nadir roll angle (in degrees) (only nadir-viewing scenes are used for focal plane calibration)	Yes
Acquisition type	Earth, Lunar, or Stellar (only Earth-viewing scenes are used for focal plane calibration)	Yes
L0Rp ID	Input L0Rp image ID	Yes
L1T image file	Name of L1T used to measure tie points	No
Acquisition date	Date of L1T image acquisition (new)	Yes
Reference image	Name of reference (DOQ) image used to measure	Yes

file	tie points	
Confidence Level	Confidence level used for outlier rejection	Yes
Fit Order	Order of Legendre fit	Yes
Number of SCAs	Number of SCAs calibrated (14) (new)	Yes
For each SCA:		
SCA Number	Number of the current SCA (1-14)	Yes
Original AT Legendre coeffs	Original along-track Legendre coefficients: a0, a1, a2	Yes
Original XT Legendre coeffs	Original across-track Legendre coefficients: b0, b1, b2	Yes
Error AT Legendre coeffs.	The computed updates to the along-track Legendre coefficients: c0, c1, c2	Yes
Error XT Legendre coeffs.	The computed updates to the across-track Legendre coefficients: d0, d1, d2	Yes
New AT Legendre coeffs	New along-track Legendre coefficients: a'0, a'1, a'2	Yes
New XT Legendre coeffs	New across-track Legendre coefficients: b'0, b'1, b'2	Yes
Pre-fit AT residual statistics	Pre-fit along-track residual mean, standard deviation, and RMSE statistics	Yes
Pre-fit XT residual statistics	Pre-fit across-track residual mean, standard deviation, and RMSE statistics	Yes
Post-fit AT residual statistics	Post-fit along-track residual mean, standard deviation, and RMSE statistics	Yes
Post-fit XT residual statistics	Post-fit across-track residual mean, standard deviation, and RMSE statistics	Yes
Number of Points	Number of tie points used for current SCA	Yes
CPF Group:		
Effective Date Begin	Beginning effective date of CPF group: YYYY-MM-DD	No
Effective Data End	Ending effective date of CPF group: YYYY-MM-DD	No
For each SCA:		
New Legendre polynomial coefficients	Nine (one per band) arrays of three along-track Legendre coefficients followed by nine arrays of three across-track Legendre coefficients.	No
Tie Point Data:	For each tie point:	
SCA Number	SCA where the tie point was measured	No
Grid Cell Column Number	Column number of the grid cell containing the tie point	No
Nominal Output Space Line	Predicted tie point output space line location	No
Nominal Output Space Sample	Predicted tie point output space sample location	No
LOS Line Error	Measured LOS error delta line (in pixels)	No
LOS Sample Error	Measured LOS error delta sample (in pixels)	No
LOS AT Error	Measured LOS error along-track delta angle	No

	(in microradians)	
LOS XT Error	Measured LOS error across-track delta angle (in microradians)	No
State Flag	Tie point state (outlier) flag	No
AT Fit Residual	Along-track fit residual (in microradians)	No
XT Fit Residual	Across-track fit residual (in microradians)	No

Table 1: Focal Plane Calibration Output Details

Accessing the Focal Plane Calibration Results in the Characterization Database
Though not part of the formal focal plane calibration algorithm, some comments regarding the anticipated methods of accessing and analyzing the individual scene focal plane calibration results stored in the characterization database may assist with the design of the characterization database (see note #3).

The database output from the focal plane alignment calibration algorithm will be accessed by a data extraction tool that queries the characterization database to retrieve focal plane calibration results from multiple scenes. The only processing required on the returned results is to compute the average "new" Legendre coefficients for each SCA across all returned scenes. The returned scene results and computed mean Legendre coefficient values will be output in a report containing a comma-delimited table of the retrieved trending results as well as the summary averages.

The geometric results would typically be queried by acquisition date and/or WRS path/row. The most common query would be based on acquisition date range, for example, selecting all of the results for a given calendar quarter:

Acquisition_Date is between 01APR2012 and 30JUN2012

The average coefficients would be calculated from the "new" Legendre coefficients for the individual scenes returned, as:

$$Coeff_{SCA,j,net} = \frac{1}{numScene} \sum_{i=1}^{numScene} Coeff_{SCA,j,i}$$

for coefficient j (j=0,1,2) for each SCA.

The query results would be formatted in a set of comma-delimited records (for ease of ingest into Microsoft Excel), one record per scene. Each record would contain all of the "header" fields written to the characterization database (items with "Yes" in the rightmost column of Table 1 above) but only the "new" Legendre coefficients for each SCA. The other fields would be retrieved using general purpose database access tools, if and when desired. A header row containing the field names should precede the database records.

Following the scene records the average Legendre coefficients should be written out in the same CPF/ODL syntax used in the report file. This output uses the same structure shown in the final row in Table 1 above, but contains the average, rather than a single scene's, Legendre coefficients.

7.2.11.8 Maturity

Most of the heritage ALIAS focal plane alignment calibration logic was reused, but the OLI version was adapted to account for the OLI sensor architecture:

1. There are 14 separate SCAs to calibrate (vs. 4 ALI SCAs).
2. The heritage Legendre polynomial order of 2 will be used for the OLI, despite the somewhat longer SCAs. This is justified by the OLI telescope design information available to date. Note that the TIRS version of this algorithm (TIRS Alignment) will require higher order Legendre terms.

7.2.11.9 Notes

Some additional background assumptions and notes include:

1. The heritage fit_polynomials procedure uses the input space reference image point locations to derive the correction polynomial coefficients. This has been changed to use the input space search image point locations (which will be very close) instead.
2. The heritage tie point selection sub-algorithm (select_corr_pts) is unnecessarily complicated. It has been reworked to use a simpler bilinear interpolation approach.
3. The trending output from this algorithm will be accessed by an analysis tool that queries the trending database to retrieve focal plane alignment results from multiple scenes. Averaging the Legendre coefficients derived from calibration scenes within a user-specified date range will smooth out residual precision correction errors, leading to a more consistent focal plane calibration solution. The analysis tool will create a report file containing a comma-delimited table of the retrieved trending results and the averaged Legendre coefficients.
4. A configuration table (system table) should be provided for each installation of the algorithm implementation to convey site-specific information such as the processing center name (used in the standard report header), the number of processors available (for parallel processing implementations), etc. This takes the place of the heritage system table which also contained certain algorithm-related parameters. Anything related to the algorithms has been moved to the CPF for LDCM. For the prototype implementation, the site-specific report header fields are provided as environment variables.
5. Heritage optional input parameters that allow the report output to be suppressed and that over-ride several of the processing parameters now provided in the CPF (minimum correlation strength, maximum correlation offset, correlation window size, and Legendre fit order) have been retained in the prototype implementation to facilitate testing, but they are not required.

7.2.12 OLI Sensor Alignment Calibration Algorithm

7.2.12.1 Background/Introduction

The OLI sensor alignment calibration algorithm uses a time sequence of the LOS model alignment trending results generated by the LOS model correction algorithm (see the LOS Model Correction ADD for details) to estimate the orientation of the OLI coordinate system relative to the spacecraft attitude determination coordinate system. This spacecraft -to-OLI alignment is one of the fundamental geometric calibration parameters stored in the Calibration Parameter File. Analyzing time sequences of measured alignment results makes it possible to smooth out random scene-to-scene pointing errors to estimate, and correct for, any underlying systematic alignment errors. The OLI sensor alignment calibration algorithm is inspired by the ALI sensor alignment algorithm used in ALIAS. Its implementation will be different, in that the ALIAS code was set up to operate on individual scene results. The heritage logic takes the precision solution output, converts it to apparent alignment errors, and then blends the individual scene results with the current best estimate of the alignment state using a Kalman filter. This approach required the scenes to be processed in time order and did not provide a view of how the apparent alignment errors varied with time, which would have made it easier to detect systematic (e.g., seasonal) effects. By retrieving and analyzing groups of individual alignment results, the OLI algorithm will make it possible to select an appropriate time window and

take all the data from that window into account when deriving the alignment calibration for that time period.

7.2.12.2 Dependencies

The OLI sensor alignment calibration algorithm assumes that the LOS model correction algorithm has populated the geometric trending database with LOS model alignment trending results.

7.2.12.3 Inputs

The OLI sensor alignment calibration algorithm uses the inputs listed in the following table. The user inputs define the parameters of a query used to retrieve the desired alignment trending data created by the LOS model correction algorithm.

Algorithm Inputs
LOS Model Correction Alignment Trending Data (from trending DB)
Precision correction reference date/time (year, day of year, hours, minutes, seconds)
Roll-pitch-yaw alignment angles (in microradians)
Ephemeris position corrections (in meters)
Alignment covariance matrix
Across- and along-track RMS GCP fit solution quality metrics (in meters)
Control type used (GLS or DOQ)
Number of control points used
GCP outlier threshold used
GCP RMS fit (in meters)
Off-nadir angle (in degrees)
Geometric characterization ID (of trended scene)
Work Order ID (of trended scene)
WRS Path/Row
User Inputs
Trending Data Query Date Range
Calibration Effective Date Range
Control Type Selection (GLS, DOQ, Both)
Control Type Weights (if Both are used) (see note 2)
Maximum off-nadir angle (in degrees) (see note 5)
Alignment Trending Flag (1 = save results)
Calibration Parameter File Name
Output Report File Name

7.2.12.4 Outputs

OLI Alignment Report (see Table 1 for details)
Standard report header fields
Control type/GCP source selected
Number of scenes analyzed
Retrieved data date range
Estimated alignment angles (roll, pitch, yaw)
Measured alignment angle RMS residuals (roll, pitch, yaw)
OLI Alignment Calibration Parameters
Alignment effective date range
ACS to OLI rotation matrix
Table of alignment trending results returned
OLI Alignment Characterization Database Output
Processing date
Processing site
Maximum off-nadir angle used

Control type/GCP source selected
DOQ vs. GLS scene weights used
Number of scenes analyzed
Retrieved data date range
Estimated alignment angles (roll, pitch, yaw)
Measured alignment angle RMS residuals (roll, pitch, yaw)
Alignment effective date range
ACS to OLI alignment matrix

7.2.12.5 Options

Control Source Selection (GLS or DOQ or Both)
Alignment Calibration Trending On/Off Switch

7.2.12.6 Prototype Code

Inputs to the executable are an ODL parameter file, an ODL calibration parameter file, and an ASCII text file that emulates the IAS trending database; outputs are an ASCII report file, an ASCII ODL-formatted CPF fragment, and trending data written to the stdout and captured in an ASCII log file.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The code units of the prototype implementation are briefly described here. Additional details are provided below for units that perform core algorithm processing logic.

aligncal.c – Main driver of the OLI sensor alignment calibration process. Also includes a utility unit to format dates as text.

get_aligncal_parms.c – Read the input ODL parameter file and pass processing parameters back to the main procedure.

query_dummy.c – A dummy unit that takes the place of a trending database query function. The dummy processes an input ASCII file called alignment_table.dat which takes the place of the trending database.

delta_date.c – Calculates the difference, in seconds, between two dates specified as year, day of year, second of day.

prec_align_to_obs.c – Processes a single trended alignment calibration record (generated by the oliprecision process) using the covariance information to combine the observed attitude and position biases into an integrated alignment error estimate.

aligncal_output.c – Subprocedure that controls the generation of the report, calibration parameter file ODL fragment, and trending data outputs.

write_report.c – Generates the output ASCII report file.

output_header.c – Creates the standard (tailored for this application) report header.

calculate_alignment_matrix.c – Constructs the attitude control system to OLI rotation matrix corresponding to a set of input roll-pitch-yaw alignment angles.

write_alignment_matrix_ODL.c – Generates the calibration parameter file ODL fragment containing the newly computed attitude-to-OLI rotation matrix.

trending_dummy.c – A dummy unit that writes the trending output from this procedure to stdout, taking the place of a trending database insertion function.

7.2.12.7 Procedure

The purpose of the sensor alignment algorithm is to use a sequence of LOS model correction solutions, including both correction parameter estimates and estimated covariance information, to estimate the underlying attitude control system (ACS) frame to OLI instrument frame alignment. A weighted least squares batch filter implementation is used to isolate the systematic alignment trend from the scene-to-scene variability of the attitude and ephemeris precision correction errors.

Unlike the other geometric correction, characterization, and calibration algorithms, the operational implementation of this algorithm will rely upon an interactive user interface that queries the geometric characterization database to retrieve a user-specified (based on date range and/or control source) set of LOS model correction alignment characterization results. The prototype implementation emulates this process by providing query parameters in an ODL parameter file and using them to filter (query) a static ASCII text file that emulates the trending database.

The algorithm uses the individual scene results returned from the database to estimate updates to the OLI alignment angles. The LOS model correction algorithm generates apparent OLI-to-ACS alignment angles each time it runs, whether on L1T product scenes using Global Land Survey (GLS) 2000 control, or on calibration scenes using digital orthophoto quadrangle (DOQ) control, based on the attitude corrections it estimates from the ground control measurements. The sensor alignment calibration algorithm analyzes these results over multiple scenes to detect the systematic trends that are used to update the ACS-to-OLI alignment estimate used in the CPF.

The sensor alignment calibration algorithm consists of five steps:

1. Allow the user to specify the date range, control source(s), and maximum off-nadir angle defining the desired range of LOS model correction alignment results.
2. Query the geometric characterization database to retrieve results meeting the specified criteria.
3. Determine the best-fit alignment angles from the individual scene results using a least squares procedure.
4. Allow the user to review the results, edit the list of input scenes used and rerun the solution, and accept or reject the final result.
5. If the result is accepted by the user, generate an output report including the list of input scenes used and the final best-fit alignment parameters, compute the corresponding ACS to OLI alignment matrix, and write out a calibration parameter group containing the alignment matrix in the format used by the CPF.

The sensor alignment calibration algorithm procedure is depicted in figure 1.

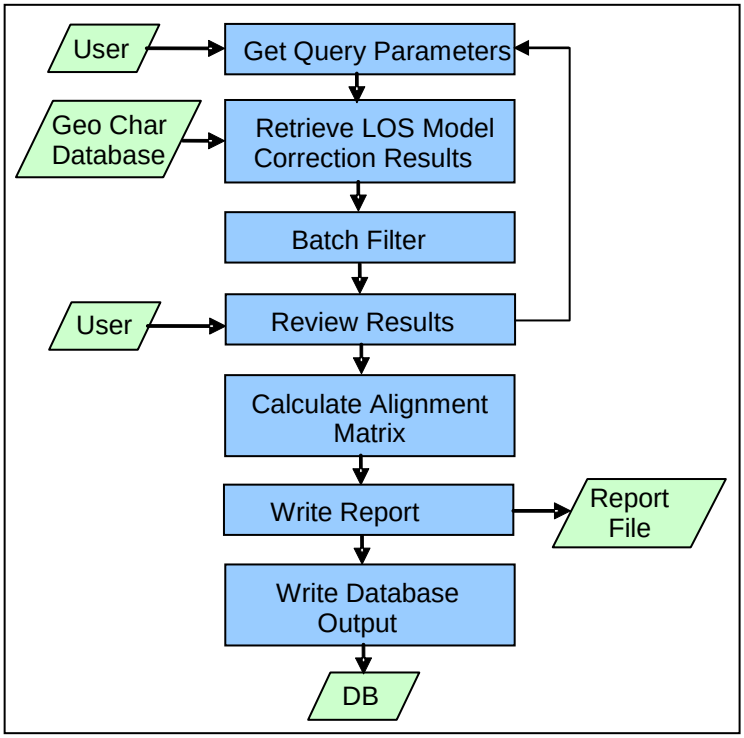


Figure 1: Sensor Alignment Calibration Algorithm Architecture

7.2.12.7.1 Step 1: Define the Data

The user provides a start and stop date to define the desired range of acquisition dates for the returned characterization data, a control type selection that makes it possible to use scenes processed with either DOQ or GLS control, or both, and a maximum off-nadir angle, in degrees, to include or exclude off-nadir acquisitions from the calibration process. The start/stop dates are inclusive. If the start date is not provided, all data acquired on or before the stop date are used. If the stop date is not provided, all data acquired on or after the start date are used. If no dates are provided, all dates are included. The DOQ/GLS/Both control selection defaults to DOQ. The maximum off-nadir angle is provided as an absolute value, i.e., only scenes between +MAXANG and -MAXANG would be included if MAXANG is the specified limit. The off-nadir angle limit defaults to 0.1 degrees to exclude off-nadir images.

7.2.12.7.2 Step 2: Retrieve the Data

The date range and control selection defined in step 1 are used to construct a database query to retrieve the desired scene records from the LOS model correction alignment table in the characterization database (see Table 3 in the LOS Model Correction ADD). All fields in this table are returned and all but the full alignment covariance matrix are displayed to the user (in step 4 below). Only the diagonal elements and the roll-Y and pitch-X elements of the covariance are displayed. The fields returned include:

1. Work order ID,
2. Geometric Characterization ID,
3. WRS path,
4. WRS row,

5. Control type,
6. Off-nadir angle,
7. Number of GCPs used,
8. GCP outlier threshold used,
9. Root-mean-square (RMS) ground control point (GCP) fit (solution quality metric),
10. Acquisition date (year, day of year) and time (hours, minutes, seconds),
11. Measured roll alignment,
12. Measured pitch alignment,
13. Measured yaw alignment,
14. Measured ephemeris (orbital) X correction,
15. Measured ephemeris (orbital) Y correction,
16. Measured ephemeris (orbital) Z correction,
17. Alignment covariance matrix:

$Cov_{roll\text{-}roll}$	$Cov_{roll\text{-}pitch}$	$Cov_{roll\text{-}yaw}$	$Cov_{roll\text{-}X}$	$Cov_{roll\text{-}Y}$	$Cov_{roll\text{-}Z}$
$Cov_{pitch\text{-}roll}$	$Cov_{pitch\text{-}pitch}$	$Cov_{pitch\text{-}yaw}$	$Cov_{pitch\text{-}X}$	$Cov_{pitch\text{-}Y}$	$Cov_{pitch\text{-}Z}$
$Cov_{yaw\text{-}roll}$	$Cov_{yaw\text{-}pitch}$	$Cov_{yaw\text{-}yaw}$	$Cov_{yaw\text{-}X}$	$Cov_{yaw\text{-}Y}$	$Cov_{yaw\text{-}Z}$
$Cov_{X\text{-}roll}$	$Cov_{X\text{-}pitch}$	$Cov_{X\text{-}yaw}$	$Cov_{X\text{-}X}$	$Cov_{X\text{-}Y}$	$Cov_{X\text{-}Z}$
$Cov_{Y\text{-}roll}$	$Cov_{Y\text{-}pitch}$	$Cov_{Y\text{-}yaw}$	$Cov_{Y\text{-}X}$	$Cov_{Y\text{-}Y}$	$Cov_{Y\text{-}Z}$
$Cov_{Z\text{-}roll}$	$Cov_{Z\text{-}pitch}$	$Cov_{Z\text{-}yaw}$	$Cov_{Z\text{-}X}$	$Cov_{Z\text{-}Y}$	$Cov_{Z\text{-}Z}$

Note that the covariance matrix can be subdivided into four 3x3 blocks:

A	**B**
$\mathbf{B}^T$	**C**

Where: **A** is the covariance of the attitude correction parameters, **C** is the covariance of the ephemeris correction parameters, **B** is the cross-covariance of the attitude and ephemeris parameters, and $\mathbf{B}^T$ is the transpose of the cross-covariance.

This formulation of the covariance matrix will be used below in combining the measured alignment and ephemeris corrections.

7.2.12.7.3 Step 3: Compute the Alignment

Computing the least squares estimate of the underlying alignment trends from the retrieved sequence of individual scene alignment measurements is complicated by the correlation between the measured angular alignment corrections and the measured ephemeris corrections. Although we do not expect to detect any systematic offset in the position bias terms (x, y, and z), they are included because of their high correlation with the attitude biases. This is reflected in the observation covariance matrix where significant off-diagonal terms will exist for X-pitch and Y-roll. Any particular LOS model correction solution will resolve the correlation between the parameters by allocating the along-track and across-track errors between the ephemeris and attitude parameters based on their a priori weights. Thus, some of the systematic alignment bias could end up allocated to the ephemeris correction terms. Over multiple precision correction solutions, the net ephemeris bias should be very close to zero. So, we use the covariance information to combine the ephemeris terms with the alignment terms to create consolidated along- and across-track corrections. In practice, since accurate GPS-derived ephemeris will be available, most of the correction will be allocated to the attitude terms in the LOS model correction solutions anyway.

Each retrieved scene provides a vector of six correction measurements:

$$[X] = \begin{bmatrix} roll \\ pitch \\ yaw \\ X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$$

where:

roll = roll alignment angle, in microradians (μrad);
pitch = pitch alignment angle (μrad)
yaw = yaw alignment angle (μrad)
X = along-track orbit position error in meters (m)
Y = cross-track orbit position error (m)
Z = radial orbit position error (m)
x = the roll-pitch-yaw 3x1 sub-vector
y = the X-Y-Z 3x1 sub-vector

The corresponding covariance matrix is also retrieved (see the LOS Model Correction ADD for a description of how these characterization data are created). It has the following structure:

$$CovX = \begin{bmatrix} Cov_{roll-roll} & Cov_{roll-pitch} & Cov_{roll-yaw} & Cov_{roll-X} & Cov_{roll-Y} & Cov_{roll-Z} \\ Cov_{pitch-roll} & Cov_{pitch-pitch} & Cov_{pitch-yaw} & Cov_{pitch-X} & Cov_{pitch-Y} & Cov_{pitch-Z} \\ Cov_{yaw-roll} & Cov_{yaw-pitch} & Cov_{yaw-yaw} & Cov_{yaw-X} & Cov_{yaw-Y} & Cov_{yaw-Z} \\ Cov_{X-roll} & Cov_{X-pitch} & Cov_{X-yaw} & Cov_{X-X} & Cov_{X-Y} & Cov_{X-Z} \\ Cov_{Y-roll} & Cov_{Y-pitch} & Cov_{Y-yaw} & Cov_{Y-X} & Cov_{Y-Y} & Cov_{Y-Z} \\ Cov_{Z-roll} & Cov_{Z-pitch} & Cov_{Z-yaw} & Cov_{Z-X} & Cov_{Z-Y} & Cov_{Z-Z} \end{bmatrix}$$

Comparing this to the A-B-C decomposition shown above, note that the "**A**" portion of the covariance matrix contains the attitude/attitude terms, the “B” portion of the matrix contains the attitude/ephemeris terms, and the "**C**" portion of the matrix contains the ephemeris/ephemeris terms.

The alignment and ephemeris corrections are combined as follows:

$$\mathbf{x}' = \mathbf{x} - \mathbf{B}\mathbf{C}^{-1}\mathbf{y}$$

where: **x**' = the consolidated along- and across-track alignment vector
x = the input alignment corrections
y = the input ephemeris corrections
B and **C** are the 3x3 covariance sub-matrices defined above.

Thus, the six LOS model correction measurements retrieved for each scene are reduced to three equivalent alignment angle observations for each scene. Consolidating the results for each scene in this manner yields a sequence of alignment angle observations:

$\mathbf{x}'_j$ where: j = 1 to N with N being the number of scenes retrieved.

Each scene observation is also assigned a (scalar) weight, w_j, based upon its control source. The DOQ and GLS weights are editable by the user, and are initially populated with default values (e.g., 50% for DOQ scenes and 50% for GLS scenes). Note that these weights are only relevant if both GLS and DOQ controlled scenes are used at the same time. The logic that enables the user to edit the weights should ensure that only numbers between 0 and 100% are allowed.

The new alignment estimate is the weighted average of these observations:

$$\theta = \frac{\sum_{j=1}^{N} w_j x'_j}{\sum_{j=1}^{N} w_j} = \begin{bmatrix} \theta_r \\ \theta_p \\ \theta_y \end{bmatrix}$$

Compute the RMS residuals:

$$RMS_r = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(roll'_j - \theta_r)^2}$$

$$RMS_p = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(pitch'_j - \theta_p)^2}$$

$$RMS_y = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(yaw'_j - \theta_y)^2}$$

The corresponding orientation matrix is:

$$M_{OLI2ACS} =$$

$$\begin{bmatrix} \cos(\theta_p)\cos(\theta_y) & \sin(\theta_r)\sin(\theta_p)\cos(\theta_y)+\cos(\theta_r)\sin(\theta_y) & \sin(\theta_r)\sin(\theta_y)-\cos(\theta_r)\sin(\theta_p)\cos(\theta_y) \\ -\cos(\theta_p)\sin(\theta_y) & \cos(\theta_r)\cos(\theta_y)-\sin(\theta_r)\sin(\theta_p)\sin(\theta_y) & \cos(\theta_r)\sin(\theta_p)\sin(\theta_y)+\sin(\theta_r)\cos(\theta_y) \\ \sin(\theta_p) & -\sin(\theta_r)\cos(\theta_p) & \cos(\theta_r)\cos(\theta_p) \end{bmatrix}$$

Take the transpose of the $\mathbf{M}_{OLI2ACS}$ matrix to compute the ACS to OLI alignment matrix used in the CPF, $\mathbf{M}_{ACS2OLI}$.

7.2.12.7.4 Step 4: Review the Results

The user is presented with a scrollable table of the individual scene results as well as the summary roll, pitch, and yaw alignment values (θ_r, θ_p, and θ_y) and the roll-pitch-yaw RMS residuals. The scene results table's columns include, in addition to the fields identified in step 2 above, the weight value, and the consolidated roll'-pitch'-yaw' values computed in step 3. The Landsat 7 Bumper Mode User Interface (BUI) is a good model for this display.

Each row in the table includes a check box or button (e.g., the BUI "Select" column) that the user can select to remove that scene/row from the alignment calculation. The user may choose to exclude scenes, based on the off-nadir angle or RMS GCP fit metrics, for example. There is also a button that the user can press to recalculate the average alignment based on the current selection of rows. The user should be able to adjust the selected scene list and recompute the average alignment as many times as desired.

A capability to plot each alignment angle is desirable but not required (see note #1). If provided it should allow the user to select which axis (roll, pitch, or yaw) to plot and then plot the corresponding consolidated angles for each selected scene on the Y axis with scene acquisition date on the X axis. It should also show the mean alignment value for that axis as a solid horizontal line or other easily identifiable symbol.

Once the user is satisfied with the alignment solution, or is ready to give up, he or she presses either the "Accept" button or the "Quit" button. The "Accept" button advances the process to step 5. The "Quit" button terminates the algorithm.

7.2.12.7.5 Step 5: Generate the Output

The sensor alignment calibration algorithm creates either two or three outputs depending upon the setting of the "Alignment Trending Flag". In all cases, a report file is generated using the input file name specified by the user, and an ODL file fragment is written out containing the ATTITUDE_PARAMETERS calibration parameter file group including the newly calculated ACS-to-OLI rotation matrix. Characterization database output is only created if the alignment trending flag is set to "Yes". The user input effective date ranges for the output calibration parameters (the ACS-to-OLI sensor alignment matrix) are embedded in the automatically generated file name used for the output ODL CPF fragment. The calibration parameter effective date range need not match the original query date range as it is often desirable to include extra data from outside the calibration time window to ensure continuity in the calibration parameters from time period to time period.

Once the solution is accepted by the user a report file is generated containing the items shown in Table 1. Note that the first 11 items in Table 1 constitute the standard report header, but that several of these fields are not applicable for a multi-scene algorithm such as sensor alignment calibration. Also note that the alignment matrix output is formatted as a CPF parameter group and includes the effective date range specified by the user. In addition to the standard report header information, the report file contains the summary alignment angles and RMS residuals, the CPF OLI alignment matrix and effective dates, and a comma-delimited table containing key fields from all the individual scene rows used in the alignment solution.

If the alignment trending flag is set, the subset of the items in Table 1 with "Yes" in the "Database Output" column are written to the characterization database.

Field	**Description**	**Database Output**
Date and time	Date (day of week, month, day of month, year) and time of file creation.	Yes
Spacecraft and instrument source	LDCM and OLI	No

Processing Center	EROS Data Center SVT (see notes #3)	Yes (see note #4)
Work order ID	Work order ID – not used for sensor alignment calibration as it operates on the results of multiple work orders.	No
WRS path	WRS path number - blank for sensor alignment cal	No
WRS row	WRS row number - blank for sensor alignment cal	No
Software version	Software version used to create report	No
Off-nadir angle	Actual maximum scene off-nadir roll angle (in degrees)	Yes
Acquisition Type	Earth viewing, Lunar, or Stellar (only Earth-viewing scenes are used for sensor alignment calibration)	No
Geo char ID	Geometric characterization trending ID – not used for sensor alignment calibration.	No
L1G image file	Not used for sensor alignment calibration	No
Acquisition date	N/A for sensor alignment calibration	No
GCP source	Ground control source used (GLS or DOQ or Both)	Yes
DOQ Weight	Weight placed on DOQ-controlled scenes (0-100%)	Yes
GLS Weight	Weight placed on GLS-controlled scenes (0-100%)	Yes
Number of scenes used	Number of scenes used in calibration	Yes
Data start date	Start date of data window used (query start)	Yes
Data stop date	Stop date of data window used (query stop)	Yes
Roll alignment	Best-fit roll alignment angle in microradians	Yes
Pitch alignment	Best-fit pitch alignment angle in microradians	Yes
Yaw alignment	Best-fit yaw alignment angle in microradians	Yes
Roll residual RMSE	RMSE of individual scene roll residuals (microradians)	Yes
Pitch residual RMSE	RMSE of individual scene pitch residuals (microradians)	Yes
Yaw residual RMSE	RMSE of individual scene yaw residuals (microradians)	Yes
Alignment effective date start	Start effective date of alignment calibration Report format: Effective_Date_Begin = "YYYY-MM-DD"	Yes
Alignment effective date stop	Stop effective date of alignment calibration Report format: Effective_Date_End = "YYYY-MM-DD"	Yes
OLI sensor alignment matrix	Best-fit ACS to OLI rotation matrix Report format: Attitude_To_OLI_Matrix = (sn.nnnnnnnnEsnn, sn.nnnnnnnnEsnn, sn.nnnnnnnnEsnn, sn.nnnnnnnnEsnn, sn.nnnnnnnnEsnn, sn.nnnnnnnnEsnn, sn.nnnnnnnnEsnn, sn.nnnnnnnnEsnn,	Yes

	sn.nnnnnnnnEsnn) where: s = sign (+/-) and n = digit	
For each scene used in the calibration:		
Work order ID	Work order ID that generated scene results	No
Geo Char ID	Geometric Characterization ID for scene	No
WRS path	Scene WRS path number	No
WRS row	Scene WRS row number	No
Control type	DOQ or GLS	No
RMS GCP fit	Root-mean-square (RMS) ground control point (GCP) fit solution quality metric	No
Acquisition date	Scene acquisition date	No
Combined roll alignment	Consolidated roll value (roll') in microradians	No
Combined pitch alignment	Consolidated pitch value (pitch') in microradians	No
Combined yaw alignment	Consolidated yaw value (yaw') in microradians	No

Table 1: Sensor Alignment Calibration Output Details

Accessing the Sensor Alignment Characterization Database

Though not part of the formal sensor alignment calibration algorithm, some comments regarding the anticipated methods of accessing and analyzing the sensor alignment results stored in the characterization database may assist with the design of the characterization database.

The database output from the sensor alignment calibration algorithm will be accessed only for purposes of reviewing the history of calibration operations. Unlike other calibration and characterization algorithms, no summary statistics are required since the sensor alignment calibration results are themselves summary statistics. Hence, a special tool to perform the query and retrieval is probably not necessary so long as basic database query capabilities are readily available.

The sensor alignment results would typically be queried by processing date, CPF effective dates, maximum off-nadir angle, and/or GCP source. The most common query would likely be a combination of GCP source and CPF effective date range, for example, selecting all of the DOQ-derived results effective for a given calendar year:

GCP_Source = "DOQ"
Effective_Start_Date is between 01JAN2013 and 31DEC2013
Effective_Stop_Date is between 01JAN2013 and 31DEC2013

The query results should be formatted in a set of comma-delimited records (for ease of ingest into Microsoft Excel), one record per scene. Each record would contain all of the fields written to the characterization database (items with "Yes" in the rightmost column of Table 1 above). A header row containing the field names should precede the database records.
y.

7.2.12.8 Maturity

Parts of the heritage ALIAS sensor alignment calibration logic were reused, but the OLI version is substantially new (but simpler).

1. A user interface to capture input from the Cal Analyst will be required, but is not provided with the prototype implementation.
2. An interface to query the geometric trending database will be required. This is emulated by a text file in the prototype implementation.
3. A capability to display the query results (as a table) to the Analyst, allowing him/her to selectively include or exclude particular entries will be required. This is not provided in the prototype implementation.
4. The updated alignment estimate computation capabilities perform the following:
    a. Merge the X and Y precision solution offsets into the pitch and roll (respectively) alignment estimates using the trended covariance information. Note that this blending was implicit in the heritage Kalman filter implementation.
    b. Fit constant (average) functions to the alignment estimates.
    c. Convert the alignment angles to the equivalent ACS-to-OLI rotation matrix for inclusion in the CPF.
5. A capability to insert the resulting alignment calibration parameters into the trending database (upon Analyst command) will also be required. This is emulated by ASCII output to stdout in the prototype implementation.

7.2.12.9 Notes

Some additional background assumptions and notes include:

1. Plotting capability that shows the individual scene alignment measurements along with the fitted constant results would also be nice.
2. Results derived from GLS control, if used at all for sensor alignment calibration, would be given lower weight than DOQ controlled scenes due to the poorer accuracy of the GLS control source as well as the additional alignment uncertainty introduced by using the SWIR1 band for GCP mensuration.
3. Configuration parameters should be provided for each installation of the algorithm implementation to convey site-specific information such as the processing center name (used in the standard report header), the number of processors available (for parallel processing implementations), etc. This takes the place of the heritage system table which also contained certain algorithm-related parameters. Anything related to the algorithms has been moved to the CPF for LDCM. This is implemented through environment variables in the prototype.
4. Most geometric characterization trending output is performed on a scene-by-scene basis so the work order ID, geometric characterization ID, and/or acquisition date fields uniquely identify the data being characterized and provide the basis for a unique database key for the trended record. In the case of sensor alignment, many input data sets are used, so provisions for a unique database record key are somewhat different. This is primarily an implementation detail, so no unique key fields are called out in the output table or Table 1 above, nor are sensor alignment calibration work order or geometric characterization IDs included in the database output. That said, the concept is that a unique sequence number would be automatically generated each time a new sensor alignment calibration record was inserted into the characterization database. In conjunction with a processing site identifier (e.g., EROS, GSFC, BATC, or SDSU), this sequence number would provide the basis for a primary key that would uniquely distinguish OLI sensor alignment calibration results generated at EROS or elsewhere.
5. Sensor alignment calibration would typically be run using the results from nadir-viewing scenes only. Since all processed scenes can produce alignment calibration results (through the LOS

model correction algorithm) the user of this application needs to be able to filter the scene results based upon the off-nadir angle field.

7.2.13 OLI MTF Bridge Characterization

7.2.13.1 Background/Introduction

The OLI Modulation Transfer Function (MTF) Characterization algorithm measures the on orbit spatial performance of the OLI using terrestrial linear targets, mostly long bridges. Historically, the Lake Pontchartrain Causeway and nearby I-10 bridge located in Louisiana have been used for this purpose since the launch of Landsat 7 in 1999. First, the Lake Pontchartrain Causeway was used to perform the across track characterization. Then the cross sectional response to the I-10 bridge was characterized and the previously determined across track component was backed out of the characterization, leaving the along track component. Both bridges cross over a large area of water providing for a fairly uniformed background against a well-defined step function.

The current algorithm generalizes the cross-sectional response approach used for the I-10 bridge to accommodate a wider variety of target geometries and locations. This makes a larger target list available for use, providing more spatial performance observations over a wider variety of acquisition conditions, hopefully leading to more robust estimates of OLI spatial performance.

The MTF bridge characterization is accomplished in four steps. The first step extracts image pixels over the bridge targets using a pre-specified UTM region of interest to identify where pixels are, and are not, to be extracted. The DN value and UTM location of each pixel are recorded so that the pixel's distance from the bridge target centerline can be calculated. The second step creates a "super-resolution" profile of the bridge by interleaving pixels, sorted by distance from target into nominally 1/8 pixel-sized bins, to create one profile sampled at 1/8 the original sample distance. The third step uses a simulated annealing method to minimize the difference between a simulated target response and the "super-resolution" data to derive OLI system transfer function (STF) parameter value estimates for the direction defined by the target's cross section. The fourth, and final, step uses target fit results from multiple cross sectional angles to estimate the underlying along- and across-track system transfer function components.

For each OLI spectral band, the characterization estimates the STF, constructs the corresponding point spread function (PSF), and edge spread function (ESF), and measures the slope of the ESF between the 40% and 60% response points. This ESF slope is the OLI spatial Key Performance Parameter (KPR) that is monitored for the operational life of the mission. Additional edge response parameters can be derived from the constructed ESF. As noted above, the STF is estimated by comparing a simulated modeled target to actual image data. The instrument model consists of optical, detector, phase, and ground sample distance (GSD) components. The along track model also contains an integration time component.

7.2.13.2 Dependencies

The OLI MTF bridge characterization algorithm assumes that a cloud free L1G image has been generated and the corresponding geometric model and grid are available. The image must be in a North-up UTM projection.

7.2.13.3 Inputs

The MTF Bridge Characterization algorithm and its component sub-algorithms use the inputs listed in the following tables. Note that some of the "inputs" are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data). MTF Bridge characterization does not normally include the OLI cirrus band.

7.2.13.3.1 MTF Extract Inputs

Algorithm Inputs	Contents
ODL Parameter File	Processing parameters
L0R File Name	Original L0R data (including metadata)
L1R File Name	Unresampled image values
Grid File Name	Maps output locations to input image space
OLI LOS Model File Name	Used to compute target range and velocity
Calibration Parameter File Name	Sensor and Earth constants
Band List	Bands to process
Target Definition File	ODL file containing bridge target parameters
Band List	Bands included in target definition
Number of Targets	Number of targets in current WRS
For each target:	Parameters provided for each target
Target Name	Identifying text name of bridge target
Water Reflectance (per band)	Water background signal (DN) level per band
Water Asymmetry (per band)	Asymmetry in water signal per band
Target Centerline UTM Coordinates	UTM X,Y defining start and end of target centerline
Number of Target Spans	Number of spans (pulses) in target
For each span:	Parameters provided for each span
Bridge Reflectance (per band)	Bridge signal (DN) level per band
Span Width (meters)	Bridge span width in meters
Span Offset (meters)	Offset (in meters) from the target centerline
Target Region of Interest (ROI)	UTM polygon(s) defining target bounds
Number of Polygons	Number of separate polygons in ROI
For each polygon:	Parameters provided for each polygon
Number of Vertices	Number of coordinates in current polygon
Coordinate List	UTM X,Y coordinates for each vertex

7.2.13.3.2 MTF Profile Inputs

Algorithm Inputs	Algorithms/sub algorithms
ODL Parameter File	Processing parameters
L0R File Name	Original L0R data (including metadata)
Band List	Bands to process
SCA List	SCAs to include (only a few will actually contain data)
Number of Profile Samples (default = 128)	Length of profile to construct
Oversampling Factor (default = 8)	Oversample factor for constructed profile
Outlier Threshold	T-Distribution Test Threshold (0.0 to 1.0)
Target Definition File Name	See MTF Extract for target definition file contents.
Pixel File Directory Name	Directory containing the MTF Extract pixel table files

7.2.13.3.3 MTF Estimate Inputs

Algorithm Inputs	Algorithm/sub-algorithms

ODL Parameter File	Processing parameters
L0R File Name	Original L0R data (including metadata)
Target Definition File Name	See MTF Extract for target definition file contents.
Profile ODL File Directory Name	Directory containing the MTF Profile output ODL files
Band List	Bands to process
Solution Method Switch	Selects Numerical Recipes (heritage) implementation (0) or GSL implementation (1) of simulated annealing
Number of Solution Iterations	Number of simulated annealing iterations to run
Max Number of Annealing Iterations	Maximum number of simulated annealing iterations to perform in each solution iteration loop (optional)
Bridge Parameter Solution Mask	Mask of 0's and 1's indicating which target parameters to adjust in each simulated annealing iteration
STF Parameter Solution Mask	Mask of 0's and 1's indicating which OLI STF parameters to adjust in each simulated annealing iteration
MTF System Table File	File containing simulated annealing initial values
Band List	Bands included in fit parameter lists
Bridge Parameter Default Values (per band)	Initial values, overridden by target definition file
Water Reflectance	Water background signal (DN) level
Water Asymmetry	Asymmetry in water signal
Ground Sample Distance (in meters)	Band nominal GSD
Bridge Reflectance	Bridge signal (DN) level (all spans)
Span Width (meters)	Bridge span width (all spans)
Span Offset (meters)	Offset from the target centerline (all spans)
STF Parameter Default Values (per band)	Initial values for OLI STF model parameters
Gaussian Optical Blur (in microradians)	Optical blur (Gaussian) dimension
XT Detector Size (in microradians)	Cross-track detector dimension
AT Detector Size (in microradians)	Along-track detector dimension
Integration Time (in milliseconds)	Detector integration time
Exponential Decay (in microradians)	ALI heritage charge diffusion distance (not used)
Model/Data Phase Shift (in meters)	Model profile/data profile registration offset
Bridge Parameter Simplex Values (per band)	Parameter variation magnitudes for 6 bridge subfields
Same 6 subfields as default values	Same units as default values
STF Parameter Simplex Values (per band)	Parameter variation magnitudes for 6 STF subfields
Same 6 subfields as default values	Same units as default values

7.2.13.3.4 MTF Perform Inputs

Algorithm Inputs	**Algorithm/sub-algorithms**
ODL Parameter File	Processing parameters
MTF Estimate Trending File Name	File containing input trending records
Spatial Performance Report File Name	Output summary report file name
Band List	Bands to process
Number of Samples	Number of samples to use in synthesizing OLI edge spread function (ESF)
Oversampling Factor	Oversample factor (relative to nominal band GSD) to used in synthesizing OLI ESF (need not be the same as the estimation oversampling factor)

MTF Estimate Trending File	File containing trending output from one or more MTF Estimate runs
Contains one trending record for each target/band analyzed	See MTF Estimate Output table for trending record field definitions

7.2.13.4 Outputs

The MTF Bridge Characterization algorithm and its component sub-algorithms create the outputs listed in the following tables.

7.2.13.4.1 MTF Extract Outputs

Pixel Table Output Files	One ASCII output file containing the pixel records for each band, polygon, and target: "TarNPolyMBandKScaL.dat" Where: N is the target number M is the polygon number K is the band number L is the SCA number
Each pixel record contains:	Pixel record fields
Band number	Band number (enumerated type, 0 to 8)
SCA number	SCA number (1 to 14)
L1R line number	L1R line number where pixel was extracted
L1R sample number	L1R sample number where pixel was extracted
UTM Y coordinate (meters)	UTM Y coordinate of pixel
UTM X coordinate (meters)	UTM X coordinate of pixel
Ground velocity (meters/second)	Pixel ground velocity (used to scale integration time)
Pixel orientation/azimuth angle (radians)	Pixel azimuth angle (from UTM grid north)
Target range (meters)	Pixel target range (used to scale angular dimensions)
Pixel signal level (DN)	Pixel image value

7.2.13.4.2 MTF Profile Outputs

Target Profile Output Files	One ODL output file containing the oversampled profile for each band and target: "Profile_TarNBandK.odl" Where: N is the target number K is the band number
Each file contains:	ODL fields contained in each output Profile file.
Target Name	Name of bridge target from target definition file
Target ID	Target number (index) within scene
Band Number	Band number for current file
Number of Samples in Profile	Length of target profile
Ground Sample Distance (in meters)	Nominal ground sample distance for current band
Oversampling Factor	Oversample factor for constructed profile
Target Azimuth Angle (in radians)	Azimuth of target relative to pixels (target orientation)
Target Range (in meters)	Average range to target (to scale angular dimensions)
Ground Velocity (in meters/second)	Average target velocity (to scale integration time)
Profile Samples (mean pixel signal values)	Constructed target profile sample sequence
Weights (profile bin standard	Standard deviations of mean profile values

deviations)	

7.2.13.4.3 MTF Estimate Outputs

MTF Bridge Characterization Report File	Text report summarizing the MTF estimation fit results
Best Fit Modeled Profile Output Files	One fitted profile output file per target and band containing comma-delimited model profile values: "ModelProfile_TarNBandK.odl" Where: N is the target number K is the band number
MTF Bridge Characterization Trending File	Fit parameter solution summary containing one record for each target and each band
Each record contains:	Pipe (|) delimited fields in each trending record
Year	Year of acquisition
Day of Year	Day of year of acquisition
WRS Path	WRS path of acquisition
WRS Row	WRS row of acquisition
Target Name	Name of bridge target
Band Number	Band number for current record
Target Orientation Angle (in radians)	Target orientation relative to image pixels
Gaussian Optical Blur (in microradians)	Optical blur (Gaussian) dimension
XT Detector Size (in microradians)	Cross-track detector dimension
AT Detector Size (in microradians)	Along-track detector dimension
Integration Time (in milliseconds)	Detector integration time
Exponential Decay (in microradians)	ALI heritage charge diffusion distance (not used)
Model/Data Phase Shift (in meters)	Model profile/data profile registration offset
Final RMS Fit	RMS fit of model to data
Ground Sample Distance (in meters)	Current band nominal GSD
Target Range (in meters)	Range to target used to scale angular units
Ground Velocity (in meters/second)	Velocity used to scale integration time
Water Reflectance	Water background signal (DN) level
Water Asymmetry	Asymmetry in water signal
Number of Spans	Number of spans in target
For each span:	Fields for each target span
Bridge Reflectance	Bridge signal (DN) level
Span Width (meters)	Bridge span width
Span Offset (meters)	I Offset from the target centerline

7.2.13.4.4 MTF Perform Outputs

OLI Spatial Performance Report File	Text report summarizing the MTF estimation fit results
For each band:	Following information is provided for each band
Summary of scenes/targets analyzed	List of each scene/target included
Final fitted 2D STF parameters	Final along- and across-track STF parameter values
Computed ES and HEE performance	Edge slope and half edge extent in each direction
Spatial performance KPR summary table	ES results and corresponding KPR penalty values
XT Synthesized ESF Profile File (per band)	One synthesized cross-track ESF output file per band containing comma-delimited normalized ESF values: "XT_ESF_BandK.odl"

	Where: K is the band number
AT Synthesized ESF Profile File (per band)	One synthesized along-track ESF output file per band containing comma-delimited normalized ESF values: "AT_ESF_BandK.odl" Where: K is the band number

7.2.13.5 Options

Heritage "Numerical Recipes in C" vs. GNU Scientific Library (GSL) simulated annealing implementation selection switch.

7.2.13.6 Prototype Code

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The following text is a brief description of the main set of modules used within the prototype with each module listed along with a very short description. It should be noted that not all library modules are referenced in the explanations below.

7.2.13.6.1 MTFEXTRACT

mtfextract – Main driver for extraction of L1R image pixels around target bridge(s) specified in input target definition file. Calls modules to retrieve ODL processing parameters, retrieve CPF parameters, extract the scene WRS path/row and retrieve the corresponding target definition file parameters, load the grid and use it to map target ROI polygons to Level-1R locations, extract image pixels from Level-1R imagery, load the model and use it to calculate pixel viewing geometry parameters, and write pixel data records to the appropriate output files.

mtf_forward_model – Uses the model to calculate the ground location of the L1R pixel.

mtf_get_ext_parms – Reads MTFEXTRACT ODL file containing processing parameters.

mtf_get_path_row – Determines the current scene WRS path/row by reading the Level 0R metadata.

mtf_get_target – Finds the SCAs that contain each target polygon, scans the polygons to identify the pixels that fall inside, extracts the associated pixel DN values, computes the pixel geometric information, and writes the resulting pixel records to an output file for the current band, SCA, target, and polygon.

mtf_get_target_descriptions – Uses the WRS path/row to construct the associated target definition file name, opens the file, and reads the target definition parameters, including the ROI polygons.

mtf_get_utm – Calculates the UTM X,Y coordinates of a L1R pixel. Also generates the target range, ground velocity, and orientation angle parameters required for the output pixel record.

mtf_init_l1r_image – Prepares the L1R image data for reading from the L1R image file.

mtf_ols_to_ils – Maps target ROI polygon vertex UTM locations to input or Level-1R locations. Uses resampling grid for mapping locations.

mtf_point_in_polygon – Performs a point-in-polygon test to determine whether a particular pixel falls inside the current polygon.

mtf_poly_sca_bounds – Calculates the bounds (in L1R line/sample space) for a particular polygon in a particular SCA to simplify subsequent search for contained pixels.

mtf_read_l1r_image – Reads a L1R image data window for a particular band and SCA from the Level-1R image file.

7.2.13.6.2 MTFPROFILE

mtfprofile – Main driver for building oversampled (super-resolution) target profiles. Calls modules for reading MTPROFILE ODL parameters, reading the target parameter file, reading the pixel table files for each band, SCA, target, and polygon, processing the pixels into distance bins to build oversampled profiles for each target, and writing out the profile data for each target to an ODL file.

mtf_calc_bin – Assign a particular pixel to a particular target profile bin based on the pixel's distance from the target centerline.

mtf_calc_view_geometry – Analyzes the pixel records for a particular target to compute summary rotation angle, range, and ground velocity parameters.

mtf_count_pixels – Determine the number of pixels in a given pixel table file. This is used to determine how much space to allocate.

mtf_filter_outliers – Uses a t-distribution test to identify and remove outlier pixels from a profile distance bin.

mtf_get_path_row – Determines the current scene WRS path/row by reading the Level 0R metadata.

mtf_get_prof_parms – Read MTFPROFILE ODL file processing parameters.

mtf_get_targets – Read the target definition file to load the target parameters.

mtf_load_pixels – Read the pixel records from a pixel table file.

mtf_process_bin – Find all the pixels contained in a given bin, remove outliers, and compute statistics.

mtf_write_profile – Writes out the oversampled profile for a target.

7.2.13.6.3 MTFESTIMATE

mtfestimate - Main driver for estimating MTF parameters from bridge oversampled (super-resolution) profiles. Calls functions to read MTF estimation processing parameters, to read target profile data files, estimate MTF parameters, and write out characterization results.

mtf_complex – Set of libraries for performing complex math operations.

mtf_fit - A package of routines that perform the best-fit analysis of analytical models of the along- and across-track system transfer functions to the bridge super-resolution profiles.

mtf_fit_gsl - A version of mtf_fit that uses GNU Scientific Library (GSL) routines to perform the simulated annealing best-fit analysis.

mtf_func_gsl - A version of the model to data fitting objective function tailored for use with the GSL simulated annealing procedure.

mtf_get_est_parms - Get the mtfestimate processing parameters from the input work order parameter ODL file and the MTF system table file.

mtf_get_sys_parms - Get the STF model parameters from the MTF system table parameter file.

mtf_get_targets – Get the target model parameters from the target definition file.

mtf_get_wo_parms - Get the processing parameters from the input work order ODL file.

mtf_model - Functions that serve as models for sensor and target (bridges). Sensor models are broken down into individual components along with one function to call all components and determine full models.

mtf_numrecipes - Source file for package of routines from "Numerical Recipes in C" that perform the heritage simulated annealing procedure.

mtf_parse_l0rname – Extracts the path/row and date from the L0R file name.

mtf_read_profile – Reads a target oversampled profile ODL file.

mtf_report_fit- Generate mtfestimate output report file.

mtf_trend_fit- Generate mtfestimate output trending data file for use by mtfperform.

mtf_write_profile - Write the best-fit model profile to a comma delimited text file.

7.2.13.6.4 MTFPERFORM

mtfperform - Main driver for calculating OLI along-track and across-track spatial performance based on the results of multiple STF model fit results from mtfestimate. Calls functions to read processing parameters, to read the mtfestimate output trending data file, estimate MTF parameters, and write out characterization results.

mtf_complex – Set of libraries for performing complex math operations.

mtf_fit_stf – Use the 1D STF fit results for multiple targets at different rotation angles (from the mtfestimate trending records) to estimate the OLI 2D (along-track and across-track) STF parameters.

mtf_get_perf_parms - Get the mtfperform processing parameters from the input ODL parameter file.

mtf_model - Functions that implement the OLI sensor model. Includes individual sensor model components along with one function to call all components and determine full model.

mtf_report- Generates mtfperform output spatial performance report file.

mtf_trendrec – Routines that read the trending records produced by mtfestimate that contain the results for particular targets.

mtf_write_esf - Write the OLI edge spread function to a comma delimited text file.

7.2.13.7 Procedure

MTF characterization generates an on-orbit estimate of the spatial (MTF and ESF) performance for the OLI instrument. The characterization estimates the OLI system transfer function (STF) and from this derives the corresponding edge spread function (ESF) from which the OLI edge slope and half edge extent key spatial performance parameters are determined. The STF is estimated by comparing a simulated modeled target to actual image data. The instrument model consists of optics, detector, sample phase, and ground sample distance components. The along track model also contains an integration time component. The bridge target model uses several long bridges at different orientations with respect to the OLI ground track, each of which is described in a target definition file. The target list includes the two bridges within WRS path 22 row 39 that have historically been used for Landsat 7 spatial performance evaluation. The current implementation can support any bridge target that can be described in a target definition file.

The MTF characterization is accomplished by running four separate routines. The first three routines are applied to OLI scenes that contain bridge targets to derive results for those targets. The fourth routine summarizes the results of multiple targets to derive the full 2D OLI STF and corresponding spatial performance. A process flow diagram showing how these routines fit together is provided in Figure 1 below. The first routine, MTFEXTRACT, extracts radiometrically corrected but unresampled image pixels over the bridge targets. These pixels are selected based upon region of interest polygons assigned to each target. These polygons can be designed to avoid problematic portions of the target area. The extracted pixel data are written to a pixel table file as pixel records that also contain geolocation and viewing geometry information. The second routine, MTFPROFILE, creates an oversampled “super-resolution” profile of the bridge target(s) by calculating each pixel’s distance from the bridge target, and sorting the pixels into distance bins that are then averaged to form an oversampled representation of the target cross-section. The bins are designed to provide an oversampled profile, nominally with a 1/8 pixel granularity. Target profiles are written out as ODL files that include pixel standard deviations as well as mean values for each distance bin. This makes it possible to weight (or ignore) individual bins based upon the pixel content (or lack thereof) in each bin. The third routine, MTFESTIMATE, uses a simulated annealing method to minimize the difference between a theoretical target response synthesized from models of the target and OLI STF, and the actual oversampled profile data.

Any one target can only provide a one-dimensional estimate of the OLI STF in the direction perpendicular to the target. Thus, each target observed provides a cross-section of the full two-dimensional OLI spatial response. The idea underlying this algorithm is that by observing targets at multiple angles, we can infer the full 2D STF from the 1D cross-sections. This is the function of MTFPERFORM, the fourth MTF characterization routine. The more target observations we have, the better this procedure should work, hence the desire to support many different targets by using a

parameterized target model. New targets can be added by simply creating an appropriate target definition file. This makes it possible to generate a richer set of observations for estimating OLI spatial performance.

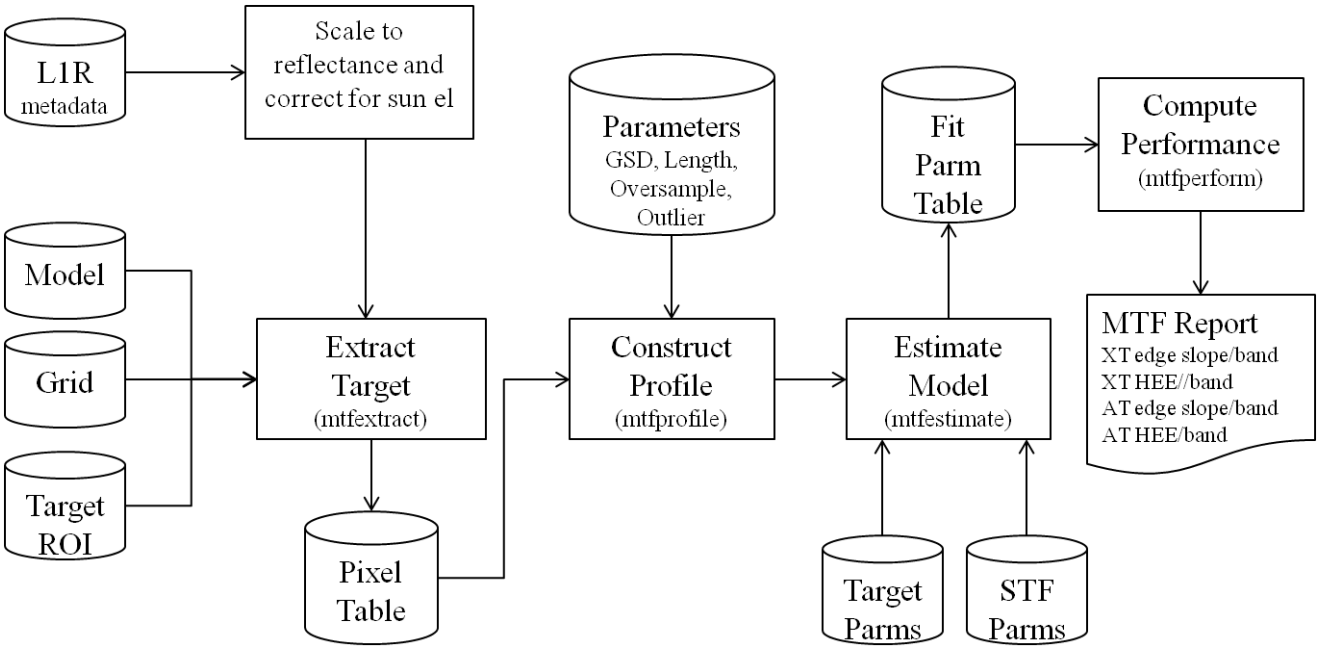


Figure 1: MTF Characterization Process Flow

The parameters required to define a bridge target include:

N_s = number of spans
For i = 1 to N_s:
w_i = span width (in meters)
x_i = span offset from target centerline (in meters)
□$_i$ = span reflectance (per band) scaled to the L1R DN range
h_i = span height above water (in meters) (not currently used)
□$_w$ = background (water) reflectance (per band) scaled to the L1R range
□□ = background asymmetry (per band) in same units as r_w
H = water WGS84 ellipsoid height (in meters)
X_s,Y_s and X_e,Y_e – UTM coordinates of target centerline endpoints
N_a = number of polygon areas in the region of interest (ROI)
For j = 1 to N_a
N_p = number of points in area (first point is not repeated at end)
For k = 1 to N_p
$X_{j,k}$,$Y_{j,k}$ = UTM coordinates of polygon area j, vertex k

These parameters are packaged in an ODL target definition file. A (simple) example of a target definition file is shown in Figure 2. Note that the file contains both the target model parameters and the target ROI.

OBJECT = MTF_Target

```
   BAND_LIST = (1,2,3,4,5,6,7,8,9)
   Number_of_Targets = 1
   Target0_Number_of_Spans = 1
   Target0_Name = "San_Mateo"
   Target0_Reflectance0 = (50.0, 100.0, 100.0, 100.0, 100.0, 35.0, 60.0, 50.0, 60.0)
   Target0_Water =          (20.0,  70.0,   70.0,   60.0,   45.0, 15.0, 15.0, 15.0, 15.0)
   Target0_Asymmetry =    (  0.0,    0.0,     0.0,     0.0,     0.0,   0.0,   0.0,   0.0,   0.0)
   Target0_Span0 = 37.165
   Target0_Offset0 = 0.000
   Target0_Coordinates = (574409.169, 4163593.065, 567700.867, 4161024.250)
END_OBJECT = MTF_Target
OBJECT = MTF_ROI
   Target0_Number_Polygons = 4
   Target0_Polygon0_Vertices = 4
   Target0_Polygon0_Coordinates = (574194.604, 4164153.388, 574623.734, 4163032.742,
                                   573849.091, 4162736.108, 573419.961, 4163856.754)
   Target0_Polygon1_Vertices = 4
   Target0_Polygon1_Coordinates = (573380.402, 4163841.605, 573809.532, 4162720.959,
                                   572241.073, 4162120.348, 571811.943, 4163240.994)
   Target0_Polygon2_Vertices = 4
   Target0_Polygon2_Coordinates = (571775.697, 4163227.114, 572204.827, 4162106.468,
                                   569041.535, 4160895.146, 568612.405, 4162015.792)
   Target0_Polygon3_Vertices = 4
   Target0_Polygon3_Coordinates = (568568.810, 4161999.098, 568997.940, 4160878.452,
                                   567915.432, 4160463.927, 567486.302, 4161584.573)
END_OBJECT = MTF_ROI
END
```

Figure 2: Sample Target Definition File MTF_Targets_044034.table

Figures 3a, 3b, and 3c show the image sampling geometry relative to the bridge target.

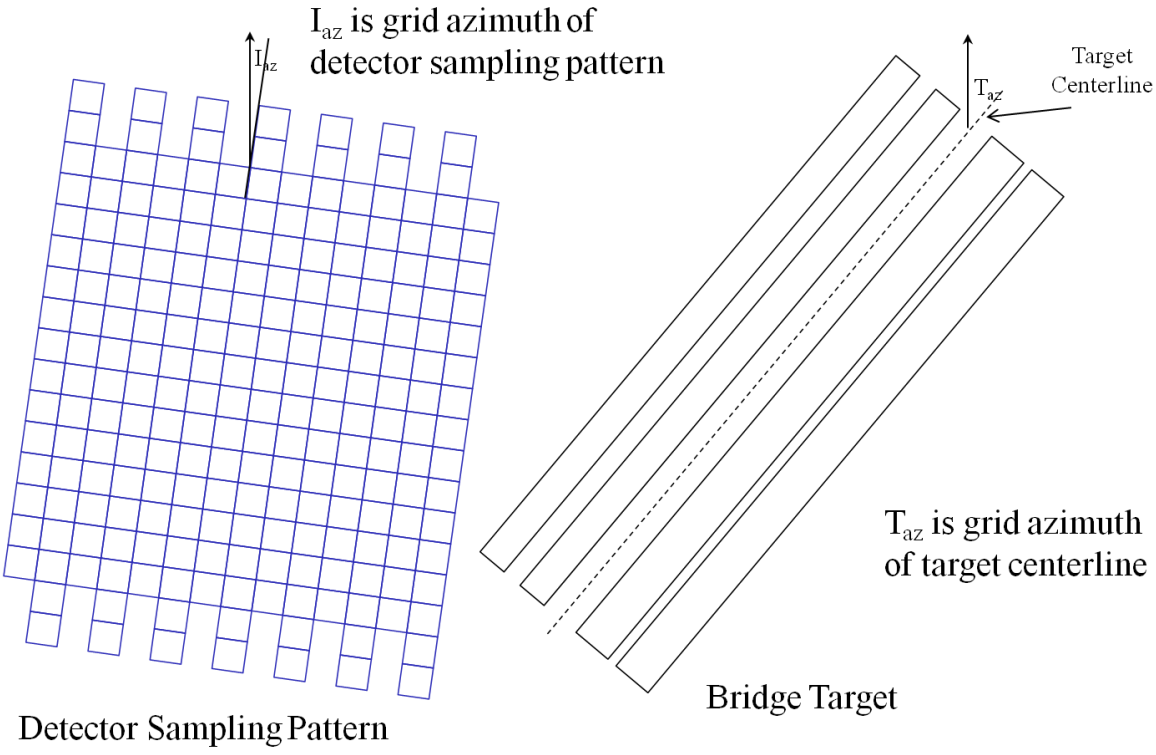


Figure 3a: UTM North-Up Detector Pattern and Bridge Target

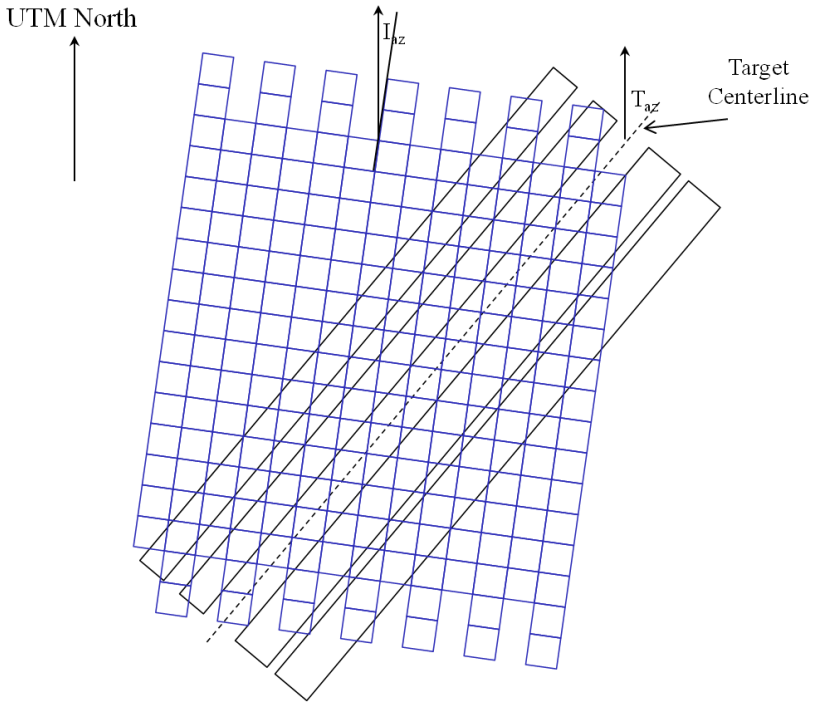


Figure 3b: Superimposed Sampling Pattern and Bridge Target

Figure 3a shows the image pixel sampling grid and the bridge target separately in a UTM north-up configuration. Figure 3b shows the image sampling pattern superimposed upon the target, and Figure 3c shows the sampled target rotated into the target orientation.

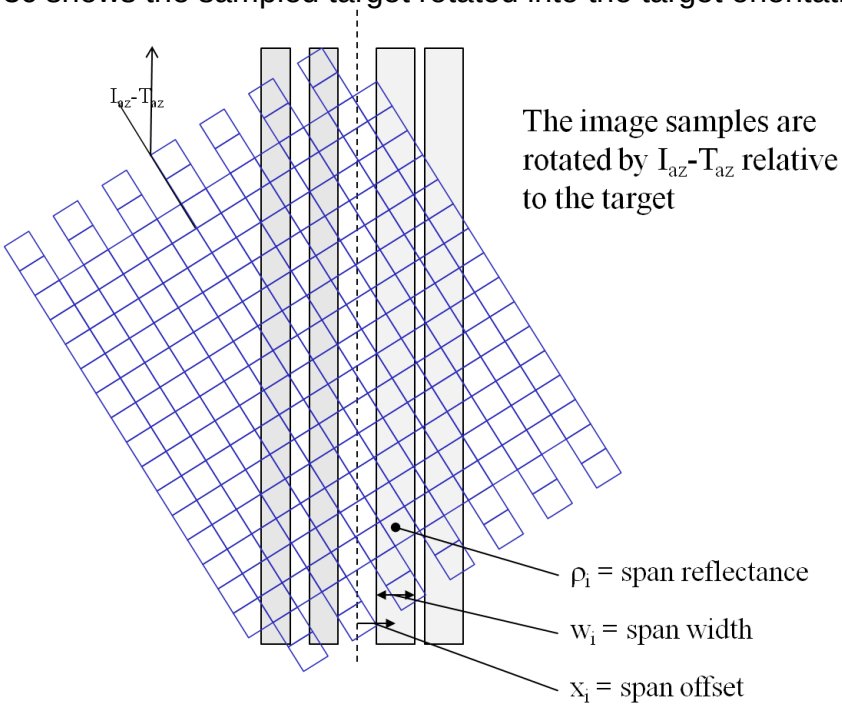


Figure 3c: Pixel Sampling Pattern and Bridge Target in Target Orientation

7.2.13.7.1 Phase 1. MTFEXTRACT: Extraction of image pixels covering the target area.

The first step in the MTF characterization process is to extract the unresampled (Level 1R) image pixels that sample the useful portion of the target area. Ideally the input image should be processed to top of atmosphere reflectance, but this is not strictly necessary. Note that not all portions of a target may be useful. For example, the Landsat 7 heritage targets, the Lake Pontchartrain Causeway and I-10 Bridge in WRS 22/39 are both double span bridges with periodic crossovers joining the spans for use as emergency vehicle turnaround areas. These crossovers are segments of the target bridge where the double-span target model is invalid. As such, the corresponding image pixels must be removed from the MTF analysis. This is controlled through the definition of the target region of interest which can be designed to contain multiple polygons so as to exclude problematic portions of the target area.

A geometrically corrected (precision and terrain) version of the OLI image is generated as an aid to correctly locating the bridge target areas. Although the corrected L1T image itself is not used, the precision model and grid used to generate it provide the information needed to project the target model into Level 1R image space. The ROI UTM coordinates are mapped back to the L1R image using the precision grid in order to locate the window for image pixel extraction. The procedure used extract the desired image pixels is as follows:

For each band to be analyzed:

Since target definition files may contain more than one target for the same path/row, the procedure loops through the targets in the target definition file:

1. Each ROI polygon for the current target is examined to determine which SCAs contain usable pixels.
   1.1 Convert each vertex in the polygon from UTM X,Y to L1G line, sample:

   l1g_line = (upper_left_y – target_y) / proj_distance_y
   l1g_sample = (target_x-upper_left_x) / proj_distance_x

   Where:
   upper_left_y = upper left Y projection coordinate of imagery (from grid)
   upper_left_x = upper left X projection coordinate of imagery (from grid)
   proj_distance_x = projection distance of imagery in X direction (from grid)
   proj_distance_y = projection distance of imagery in Y direction (from grid)
   target_x = UTM X location of ROI polygon vertex (from target definition file)
   target_y = UTM Y location of ROI polygon vertex (from target definition file)
   l1g_line = L1G image line location corresponding to ROI vertex
   l1g_sample = L1G image sample location corresponding to ROI vertex

   1.2 For each SCA, map the l1g_line and l1g_sample output space (L1G) coordinates through the precision grid (using ols2ils) to obtain the corresponding input space (L1R) l1r_line and l1r_sample coordinates.

1.3 If any vertices fall inside the SCA, mark this SCA as active for the current target.

2. For each SCA that contains valid pixels:
   2.1 Open the L1R image for the current band, SCA.
   2.2 Open an output pixel table file for the current target, polygon, band, SCA.
   2.3 Compute the minimum bounding rectangle (in L1R line/sample coordinates) of the current polygon in the current SCA.
   2.4 Loop through the MBR line/sample range:
   2.5 Perform a point-in-polygon test to determine if the current L1R line/sample falls inside the polygon.
   2.6 If it does not, go to the next point.
   2.7 Otherwise, use the model to compute the ground latitude/longitude of the current pixel using the target elevation as the height coordinate.
   2.8 Convert the latitude/longitude to UTM X/Y.
   2.9 Subtract 1 from the line number and repeat steps 2.7 and 2.8 above to compute the UTM X/Y coordinates of the same detector in the previous line (X_1,Y_1).
   2.10 Add 1 meter to the target elevation and repeat steps 2.7 and 2.8 above to compute the UTM X/Y coordinates of a point 1 meter higher along the same line of sight. Subtract the original X/Y coordinates to yield the change in X and Y per unit height: dX/dh and dY/dh
   2.11 Compute the grid azimuth of the pixel grid:
   $I_{az} = \text{atan2}(X_1 - X, Y_1 - Y)$
   2.12 Compute the ground velocity:
   $V_g = \text{sqrt}((X_1 - X)^2 + (Y_1 - Y)^2) / \text{line\_time}$
   2.13 Compute the distance from the sensor to the target using the spacecraft position **P** (returned by the forward model) and the computed ground point position **X**:
   $D = | \mathbf{X} - \mathbf{P} |$
   2.14 Compute the line-of-sight zenith angle in the direction perpendicular to the target, using the target grid azimuth computed from the target centerline coordinates T_{az}:
   $\beta = \text{atan}(dX/dh * \cos(T_{az}) - dY/dh * \sin(T_{az}))$
   2.15 Compute the cosine of the OLI viewing angle, which is just the Z component of the OLI line-of-sight (returned by oli_findlos):
   $\cos(\alpha) = los.z$
   2.16 Compute the effective target range for scaling angular units (e.g., IFOV) to ground meters (see Figure 4):
   $R = D \cos(\alpha) / \cos(\beta - \alpha)$
   2.17 Write the band, SCA, L1R line, L1R sample, UTM X, UTM Y, V_g, I_{az}, R, and the pixel DN value to the output pixel table file.

Note that a separate output pixel table file is created for each target, polygon, band, and SCA that contains usable data. Sample pixel table output is shown in Table 1.

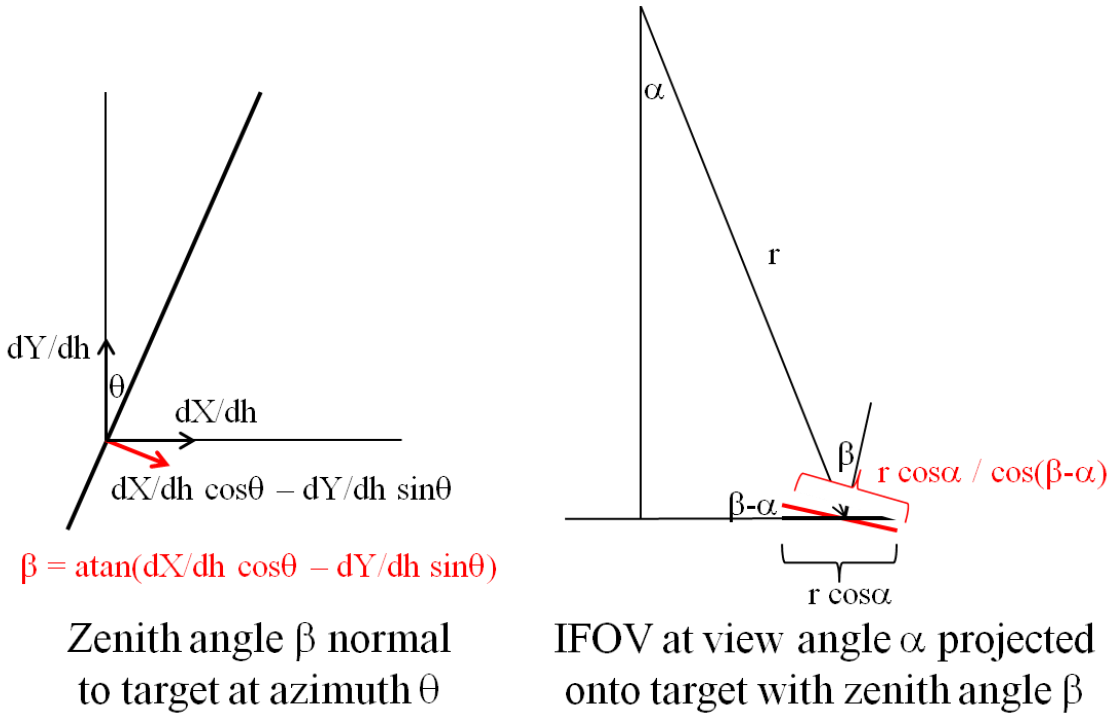


Figure 4: Geometry of Scaling IFOV in Angular Units to Ground Meters

Band	SCA	Line	Sample	UTM_Y	UTM_X	Velocity	Azimuth	Range	DN	
3	1	4020.000	403.000	2900701.181185	424260.429036	6834.884661	0.220284	706065.972359	46
3	1	4020.000	404.000	2900753.848742	424301.915342	6834.886712	0.220284	706066.347523	48
3	1	4020.000	405.000	2900689.872665	424317.441281	6834.887905	0.220284	706065.348473	45
3	1	4021.000	403.000	2900672.948225	424254.048016	6833.106791	0.222279	706065.949158	46
3	1	4021.000	404.000	2900725.615771	424295.534335	6833.108760	0.222278	706066.324314	46
3	1	4021.000	405.000	2900661.639691	424311.060258	6833.110150	0.222279	706065.325258	46
3	1	4021.000	406.000	2900714.307151	424352.545791	6833.112119	0.222278	706065.701224	47
3	1	4021.000	407.000	2900650.331112	424368.070926	6833.113508	0.222279	706064.701945	46
3	1	4021.000	408.000	2900702.998484	424409.555673	6833.115477	0.222278	706065.078712	44
3	1	4021.000	409.000	2900639.051627	424425.086519	6833.116867	0.222279	706064.079550	44
3	1	4021.000	410.000	2900691.689772	424466.563981	6833.118836	0.222278	706064.456787	44
3	1	4021.000	411.000	2900627.742955	424482.094039	6833.120225	0.222279	706063.457398	44
3	1	4021.000	412.000	2900680.381013	424523.570715	6833.122195	0.222278	706063.835436	45
3	1	4021.000	413.000	2900616.434236	424539.099984	6833.123583	0.222279	706062.835829	45
3	1	4021.000	414.000	2900669.072208	424580.575875	6833.125553	0.222278	706063.214678	43
3	1	4021.000	415.000	2900605.119170	424596.132535	6833.126943	0.222279	706062.214534	43
3	1	4021.000	416.000	2900657.763357	424637.579462	6833.128911	0.222278	706062.594505	44

Table 1: Sample Pixel Table Output

7.2.13.7.2 Phase 2. MTFPROFILE: Create the oversampled target profile(s)

The second step in the MTF characterization process is to produce an oversampled "super-resolution" profile of each target for each image band. This is done by calculating the distance from each sample to the centerline of the target (preserving sign based on which side of the target the sample falls on), rescaling this distance from ground meters to profile samples by dividing by the

nominal image GSD and multiplying by the desired oversampling factor, collecting all of the samples that fall in each profile bin and computing a bin average DN value (after removing outliers), ordering the bin averages to form the output profile, and computing target average orientation angle, range, and ground velocity values for subsequent use mtfestimate.

The profile construction procedure is as follows:

Load the target definition file for the current path/row.

For each target:
Calculate the grid azimuth of the target:

$T_{az} = \text{atan2}(X_e – X_s, Y_e – Y_s)$

Where: X_s, Y_s are the UTM coordinates of the target centerline start point
X_e, Y_e are the UTM coordinates of the target centerline end point

For each band:

1. Read all of the pixel records over all SCAs and polygons, for this target and band.
2. For each pixel record, calculate the perpendicular distance to the target:
   a. Construct the vector from the centerline start point to the pixel:
      $v_i = [(X_i – X_s) \quad (Y_i – Y_s) \quad 0]^T$
   b. Construct the target centerline vector:
      $v'_t = [(X_e – X_s) \quad (Y_e – Y_s) \quad 0]^T$
   c. Normalize to form the corresponding target centerline unit vector:
      $v_t = v'_t / | v'_t |$
   d. Define the Z direction unit vector:
      $v_z = [0 \quad 0 \quad 1]^T$
   e. Calculate the distance by taking the cross product of the pixel vector with the centerline unit vector, and dotting the result with the Z direction unit vector:
      $D_i = v_z \bullet (v_i \times v_t)$
      The dot product extracts the Z component of the cross-product vector.
      Note that this calculation preserves the sign of the distance with points to the “right” of the target having positive distances and points to the “left” of the target having negative distances. Clearly, the sign convention depends upon the definition of the centerline start and end points.
   f. Convert the signed distance to a bin number:
      $B_i = \text{round}(D_i * O_s / GSD + N/2)$

   Where: O_s is the specified oversampling factor
   GSD is the nominal GSD for this band
   N is the specified length of the profile (number of bins)
3. For each pixel record, compute the target orientation relative to the pixel:
   $\theta_i = I_{az} - T_{az}$
4. For each bin:
   a. Calculate the mean and standard deviation of the pixel DN values in the bin and perform a t-distribution outlier rejection test based upon the user-specified outlier threshold.
      i. If the DN farthest from the mean exceeds the t-distribution threshold, flag it as an outlier and recomputed the statistics without that point.
      ii. Continue removing the farthest point until all points pass the t-distribution outlier test.

b. Record the DN mean and standard deviation for all non-outlier points in the bin.

5. Compute the average, θ_{mean}, of the θ_i target orientation angle values for all non-outlier points across all bins.
6. Compute the average R_{mean} of the R_i (range) values for all non-outlier points across all bins.
7. Compute the average V_g of the V_i (ground velocity) values for all non-outlier points across all bins.
8. Write the output profile ODL file containing target name (from the target definition file), the target ID (number), the band number, the profile length (N), the oversampling factor (O_s), the nominal GSD, the target azimuth (θ_{mean}), the target range (R_{mean}), the ground velocity (V_g), the N-element profile data vector of mean DN values, and the N-element vector of bin standard deviations.

Note that this procedure creates one output profile ODL file for each target and each band. A sample profile ODL file is shown in Figure 5.

```
OBJECT = TARGET_PROFILE
 TARGET_NAME = "King_Fahd_Causeway_W"
 TARGET_ID = 0
 BAND = 4
 NSAMP = 128
 GSD = 30.000
 OVERSAMP = 8
 TARGET_AZIMUTH =  1.515855155
 TARGET_RANGE = 706080.268
 GROUND_VELOCITY = 6832.106
 PROFILE = ( 555.888889, 579.500000, 573.000000, 574.000000, 575.416667, 564.200000, 576.833333, 569.300000, 571.100000, 567.222222, 565.333333,
       572.800000, 574.000000, 576.833333, 571.000000, 574.400000, 579.700000, 571.000000, 567.909091, 576.666667, 579.700000, 576.916667,
       574.400000, 567.600000, 584.800000, 569.300000, 569.666667, 574.272727, 571.333333, 581.400000, 565.900000, 567.700000, 564.200000,
       581.400000, 572.500000, 576.833333, 575.727273, 588.333333, 577.500000, 577.900000, 571.100000, 563.714286, 578.000000, 569.454545,
       574.000000, 559.000000, 554.000000, 553.875000, 560.800000, 560.181818, 566.625000, 618.500000, 615.750000, 647.300000, 701.000000,
       709.500000, 763.625000, 830.583333, 888.500000, 921.750000, 973.500000, 988.400000, 1041.800000, 1062.600000, 1015.333333, 1038.444444,
       985.083333, 928.833333, 852.500000, 934.500000, 801.300000, 761.600000, 695.250000, 640.333333, 653.888889, 617.333333, 584.800000,
       562.250000, 562.500000, 565.900000, 558.166667, 578.166667, 576.666667, 580.666667, 574.400000, 574.500000, 575.375000, 582.666667,
       585.181818, 585.500000, 585.333333, 582.333333, 583.285714, 572.700000, 581.200000, 578.100000, 582.666667, 575.727273, 585.500000,
       581.400000, 565.333333, 578.285714, 579.700000, 586.916667, 571.000000, 582.666667, 580.545455, 577.900000, 571.000000, 576.666667,
       583.428571, 575.250000, 577.181818, 572.800000, 588.333333, 572.888889, 579.600000, 572.800000, 578.000000, 578.428571, 575.333333,
       573.916667, 572.888889, 562.500000, 578.666667, 576.300000, 575.333333, 557.400000 )
 SIGMAS = ( 10.215729, 12.020815, 21.765799, 24.162330, 20.926097, 15.205262, 14.288690, 12.543701, 18.131924, 7.496295, 9.814955, 18.902087,
       19.249557, 17.049838, 12.020815, 14.223220, 21.929432, 11.333333, 16.688047, 9.814955, 18.672916, 23.635329, 10.751744, 14.223220, 14.584238,
       12.543701, 18.597817, 20.164776, 30.022214, 24.604426, 11.474125, 17.795443, 15.205262, 15.598077, 18.583962, 19.894418, 17.326805, 30.022214,
       12.961481, 14.548387, 16.264822, 9.086882, 19.735754, 9.169118, 17.832555, 24.289916, 24.041631, 25.892291, 18.274451, 15.715077, 33.027856,
       25.890796, 29.942748, 26.038646, 50.497525, 24.748737, 32.539811, 39.083729, 58.670885, 69.244391, 59.107247, 63.972564, 44.534132, 43.169434,
       43.775945, 48.179641, 59.689284, 58.203145, 75.500237, 48.790368, 49.647759, 55.869292, 48.392571, 46.112182, 40.489025, 46.245065, 30.367380,
       29.158434, 12.020815, 32.460061, 20.831939, 15.467463, 9.814955, 19.538424, 17.557525, 22.618822, 15.352408, 26.407070, 17.051793, 21.952428,
       17.803643, 9.814955, 13.136644, 12.543701, 8.778762, 21.926392, 26.407070, 15.569784, 19.332027, 18.566397, 9.814955, 13.375528, 16.865810,
       17.333406, 8.500000, 26.407070, 20.992206, 16.609569, 16.027754, 9.814955, 19.303343, 12.814232, 13.753677, 15.127605, 17.502381, 13.289511,
       14.645439, 17.119190, 19.735754, 13.709573, 16.576727, 16.115821, 17.919573, 12.020815, 15.215124, 18.366939, 16.576727, 14.223220 )
END_OBJECT = TARGET_PROFILE
END
```

Figure 5: Sample Output Profile ODL File

7.2.13.7.3 Phase 3. MTFESTIMATE: Fit target and STF models to profile data

The MTFESTIMATE routine consists of an initialization stage and a model parameter estimation loop. During the initialization phase, processing parameters are read from the input ODL parameter file, initial STF model parameters are read from a system table file, and the target model parameters are read from the target definition file. The processing parameters include arrays of flags that determine which variables to solve for in each iteration of the solution procedure. In addition to the STF model

parameters, the system table file includes simplex bounds that control the behavior of the simulated annealing procedure that is used to solve for the STF and target model parameters. The output report and trending data files are also created during the initialization phase.

The model parameter estimation loop cycles through each target in the target definition file. For each target, the selected spectral bands are analyzed one-by-one. The MTFPROFILE routine will have created an oversampled target profile for each target/band combination. The best fit model parameters for each band and each target are found using a simulated annealing approach. Two user selectable implementations of the simulated annealing procedure are included, one provided by the GNU Scientific Library (GSL) and the other a modified version of the simplex downhill method described in “Numerical Recipes in C”. The latter is the heritage Landsat 7 implementation. The GSL implementation is substantially faster and appears to be robust, but our lack of experience with this software argued for retaining the heritage method as an option.

The simulated annealing algorithm allows for the solution of multi-variable non-linear functions. Multiple iterations of the annealing process are performed. Each iteration adjusts a user specified subset of the model parameters to best fit the oversampled target profile data. There are two models of functions involved in the process, one describing the bridge target and one describing the imaging system transfer function. Both the target and system (STF) models are formulated in the frequency domain. The target and system functions are evaluated in the frequency domain. The target function is then multiplied by the system transfer function in the frequency domain, producing a frequency domain representation of the simulated target profile. The inverse Fourier Transform is found of the simulated target response. This result is then compared against the oversampled target data profile generated in phase 2. The root mean squared error between the synthesized data profile, produced from the models, and the measured data is used as the metric for the objective function minimization process in the simulated annealing procedure.

The Target Model

The bridge targets consist of one or more spans, each modeled as a rectangular pulse of a specified width at a specified offset from the target centerline, against a background (water) component. The background is allowed to be different on the two sides of the target by including an asymmetry term in the background model. The combined model is then the summation of the contributions from the individual bridge target spans and the background component. These target model components can be described in the frequency domain as follows:

$$\text{Span i:}\quad \mathrm{S_i}(\omega) = (\rho_i - \rho_W) * \left(\frac{w_i * O_s}{GSD} \right) * \mathrm{sinc}\left(\omega \frac{w_\mathrm{i}}{2} \right) * e^{j\omega x_i}$$

$$\text{Background}: \mathrm{b}(\omega) = \mathrm{N} * \rho_\mathrm{w} * \delta(\omega) + \Delta\rho * \frac{N}{2} * \mathrm{sinc}\left(\omega \frac{D}{4} \right) * e^{-j\omega \frac{D}{4}}$$

$$\delta(\omega) = \begin{Bmatrix} 1 \ \text{ if } \omega = 0 \\ 0 \ \text{otherwise} \end{Bmatrix}$$

Where:
ω = spatial frequency in radians per meter, evaluated at:
$\omega = O_s * 2\square / GSD * i / N$ for i = -N/2 to (N/2 - 1)
w_i = bridge span i width (in meters)
x_i = span i offset from centerline (in meters)
O_s = oversampling factor
GSD = nominal image ground sample distance (in meters)
N = target sequence length in samples
D = target sequence length in meters = $N * GSD / O_s$
ρ_i = span i reflectance
ρ_w = background (water) reflectance
$\Delta\rho$ = background reflectance asymmetry (difference between bridge sides)
sinc = sin(x)/x
$j = \sqrt{-1}$

The total target model is then the sum of the components above:

$$\text{Target}: \mathrm{T}(\omega) = \sum_{i=1}^{N_s} \mathrm{S}_\mathrm{i}(\omega) + b(\omega)$$

Where:
N_s = the number of spans in the target bridge model

The variable target model parameters are:
the w_i span width values (one per span),
the x_i span offset values (one per span),
the ρ_i span reflectance values (one per span),
the ρ_w water reflectance value, and
the $\Delta\rho$ water reflectance asymmetry value.
The target physical dimensions, w_i and x_i, are held fixed while the reflectance values are allowed to vary and are typically input into the simulated annealing simplex solution as unknown variables to solve for.

The System Transfer Function Model
The OLI system transfer function model contains optical and detector components. The optical model includes both exponential and Gaussian components. The Gaussian component models optical blur and is the primary element of the optical sub-model. The exponential term was used in modeling the Advanced Land Imager (ALI) system transfer function and is included as a heritage element of the model, but is not expected to be used for OLI STF modeling. The detector model includes both detector dimension and integration time components. All of the model sub-components are included in both the along-track and across-track STF models except for the integration time component, which is only part of the along-track model.

The optical component models are formulated in the frequency domain as follows:

$$\text{Optical (Gaussian):}\quad \mathrm{O}(\omega) = e^{-\frac{\omega^2 * \sigma^2}{2}}$$

Where:
σ=optical blur radius (in microradians) scaled to ground meters using R_{mean}, the target range parameter determined by MTFPROFILE

$$\text{Exponential:}\quad \mathrm{C}(\omega) = e^{-|\omega * d|}$$

Where:
d=exponential decay distance (in microradians) scaled to ground meters using R_{mean}, the target range parameter determined by MTFPROFILE

The detector dimension component model is formulated in the frequency domain using a sinc function:

$$\text{Detector:}\quad \mathrm{D}(\omega) = \mathrm{sinc}\left(\frac{\omega * r}{2}\right)$$

Where
r = detector angular dimension (in microradians) scaled to ground meters using R_{mean}, the target range parameter determined by MTFPROFILE

The integration time model is also formulated using a sinc function. In this case the pulse dimension is the integration time scaled by the ground velocity, V_g.

$$\text{Integration:}\quad \mathrm{I}(\omega) = \mathrm{sinc}\left(\frac{\omega * \tau * V_g}{2}\right)$$

Where
τ = detector integration time (in milliseconds) scaled to ground meters using V_g, the ground velocity parameter (in km/sec or m/msec) determined by MTFPROFILE

The combined across-track STF model is then the combination of the optical, exponential, and detector dimension components.

Cross-Track STF: $STF_x(\omega) = O(\sigma_x,\omega)\ C(d_x,\omega)\ D(r_x,\omega)$

Where:
σ_x = cross-track component of Gaussian blur
d_x = cross-track component of exponential decay

r_x = cross-track detector dimension

The along-track STF model is similar but includes the integration time component:

Along-Track STF: $STF_a(\omega) = O(\sigma_a,\omega)\ C(d_a,\omega)\ D(r_a,\omega)\ I(\tau,\omega)$

Where:
σ_a = along-track component of Gaussian blur
d_a = along-track component of exponential decay
r_a = along-track detector dimension
τ = detector integration time

Note that the integration time parameter has no directional index since it applies only in the along-track direction.

Synthesizing the Target Response

The modeled target response is synthesized from the component target and STF models by multiplying the system transfer function by the target model, computing the corresponding space domain representation by taking the inverse Fourier transform, and then comparing the resulting modeled target response to the measured target profile. There are two problems with this. The first is that the target model is a one-dimensional model that applies in the direction perpendicular to the bridge target whereas the STF model is a two-dimensional model that is separable into along- and across-track terms. We must determine the effective one-dimensional response of the 2D STF in the direction perpendicular to the bridge.

To do this, consider the relative rotation angle between the detector sampling pattern and the bridge target:

$\theta = I_{az} - T_{az}$

Where:
I_{az} is the detector sampling pattern UTM grid azimuth
T_{az} is the target UTM grid azimuth

The average θ angle for the target was computed by MTFPROFILE and is contained in the input oversampled profile ODL file. The STF model is formulated relative to the cross-track (ω_x) and along-track (ω_a) directions, so we express this coordinate system in terms of the the target longitudinal (ω_{long}) and transverse (ω_{trans}) directions (see Figure 6):

$\omega_x = \omega_{trans} \cos\theta - \omega_{long} \sin\theta$
$\omega_a = \omega_{trans} \sin\theta + \omega_{long} \cos\theta$

In the longitudinal direction we are interested only in the DC frequency term since we seek to integrate away this direction. So, setting $\omega_{long} = 0$ and $\omega_{trans} = \omega$, we can express ω_x and ω_a in terms of the transverse direction, which is the axis of the target model:

$\omega_x = \omega \cos\theta$
$\omega_a = \omega \sin\theta$

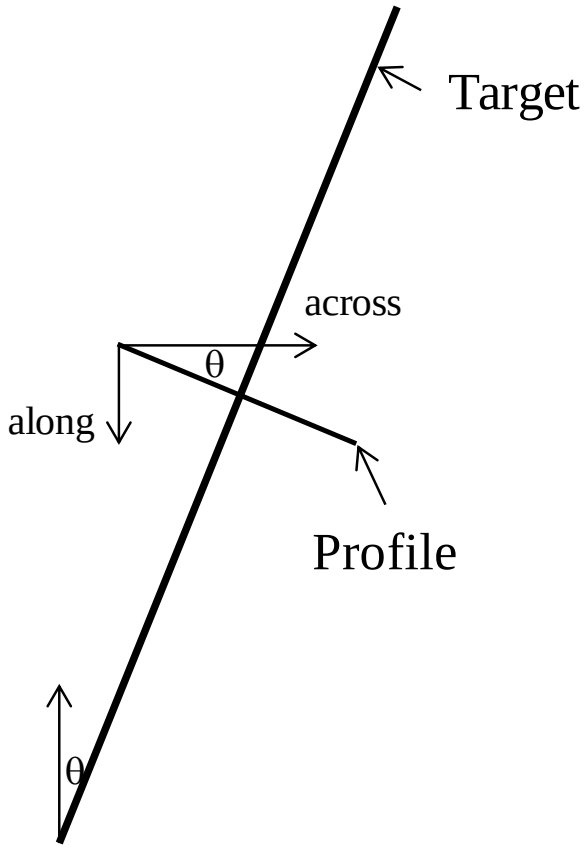


Figure 6: Bridge Cross-Section Profile at Angle Relative to Across-Track

These expressions can then be substituted into the STF model to yield the effective 1D STF in the target profile direction.

$STF_{trans}(\omega) = STF_x(\omega \cos\theta)\ STF_a(\omega \sin\theta)$

The second problem mentioned above is that we need an additional model component to register the synthesized analytical target response to the actual profile data, in the transverse (across-bridge) direction. This registration component takes the form of a phase shift:

$$\text{Phase:}\quad P(\omega) = e^{-j\omega p}$$

Where:
p = scene dependent phase shift required to register the model to the profile data

The combined model used to synthesize the target response includes the transverse (1D) STF model, the target model, and the phase adjustment:

$M(\omega) = STF_x(\omega \cos\theta)\ STF_a(\omega \sin\theta)\ T(\omega)\ P(\omega)$

Substituting in the appropriate STF model components from above:

$$M(\omega) = O(\sigma_x, \omega\cos\theta)\ D(r_x, \omega\cos\theta)\ C(d_x, \omega\cos\theta)$$
$$O(\sigma_a, \omega\sin\theta)\ D(r_a, \omega\sin\theta)\ C(d_a, \omega\sin\theta)\ I(\tau, \omega\sin\theta)\ T(\omega)\ P(\omega)$$

For the time being we also assume that $\sigma_x = \sigma_a = \sigma$ so that:

$$O(\sigma_x, \omega\cos\theta)\ O(\sigma_a, \omega\sin\theta) = O(\sigma, \omega)$$

Substituting yields:

$$M(\omega) = O(\sigma, \omega)\ D(r_x, \omega\cos\theta)\ C(d_x, \omega\cos\theta)$$
$$D(r_a, \omega\sin\theta)\ C(d_a, \omega\sin\theta)\ I(\tau, \omega\sin\theta)\ T(\omega)\ P(\omega)$$

We will undo this simplifying assumption in phase 4: MTFPERFORM, but for this phase of the procedure we will transform $M(\omega)$ to the space domain as:

$$M(x) = F^{-1}\{ M(\omega) \}$$

And best fit M(x) to the measured profile data by adjusting the model parameters.

For the OLI system transfer function, the detector dimensions (r_x, r_a) and integration time (τ) are known to a high degree of precision, and the exponential decay model terms (d_x, d_a) are set to zero by definition. This only leaves the (single) Gaussian optics (σ) parameter, the phase (p) parameter, and the target model parameters as parameters subject to variation in the simulated annealing solution procedure.

The MTFESTIMATE model parameter solution procedure loops through each target specified in the target definition file. For each target we proceed as follows:

1. Preparing data for MTF estimate

   1.1 Set up solution and model parameters. Initial values for both the bridge target, and the along track and across track STF parameters are read from the MTF system table file. The initial “step” values for the simplex routine are also loaded from the system table. The generic default target model parameters are then replaced with the bridge-specific values from the target definition file.

   Loop on bands solving for the set of target and STF model parameters specified in the input ODL parameter file.

2. Read target profile produced from MTFPROFILE. This includes the target orientation angle, range scaling and ground velocity parameters in addition to the profile itself, as shown in Figure 5 above.

3. Perform parameter estimation.

3.1 Refine the bridge model parameter for background (water) by averaging DN values in profile generated in MTFPROFILE.

Loop on number of iterations of the simulated annealing process that is to be performed.

3.2 Set up bridge and system models needed for simulate annealing. Determine the model parameters to be solved for each iteration of the simulated annealing process. These are specified as input parameters. Set the step size to be used for each parameter in the simulated annealing process.

3.3 Determine the RMS fit for the initial model parameters.

The RMS is determined from the following steps:

3.3.1 Calculate the frequency domain response of the bridge target using the bridge model parameters (See The Target Model).

3.3.2 Calculate the frequency domain response of the system using the system model parameters (See The System Transfer Function Model).

3.3.3 Combine/multiply the target and the bridge responses, taking into account the phase shift parameter:

$$M(\omega) = STF(\omega) * T(\omega) * e^{-j*\omega*phase}$$

Where
STF(ω) = system modeled transfer function
T(ω) = bridge target modeled transfer function
M(ω) = synthesized bridge profile produced from modeled bridge target and modeled system transfer function
phase = phase shift required to align model with data

3.3.4 Compute the inverse FFT of M(ω).

3.3.5 Calculate RMS between image data profile (produced from MTF profile) and modeled profile:

$$RMS = \sum_{n=0}^{NPTS} \left(M(n) - Image(n)\right)^2$$

Where
M = spatial domain transfer function of modeled target profile
Image = bridge profile from image data produced from MTFPROFILE
NPTS = number of points in MTF profile

3.4 Perform simulated annealing.

3.5 Check to see if optical and detector model parameters are positive. If any one of the parameters is not positive, take the absolute value of the parameter.

3.6 If necessary adjust the phase of the system transfer function to be within one half of a cycle of the bridge ground sample distance:

The solution of system transfer function returns a phase. This phase is periodic with respect to the number of over-sampled bins and the ground sample distance (N_{over}=N/2 for a profile length of N). Therefore the phase can be adjusted by multiples of N_{over}*GSD (ground sample distance).

while (PHASE >= N_{over}*GSD)
PHASE -= 2*N_{over}*GSD
while (PHASE < -N_{over}*GSD)
PHASE += 2*N_{over}*GSD

Where: PHASE = solution for phase component of system model

4. Write the results for this band to the report file.

4.1 Generate the final modeled spatial domain target response (profile) using the model parameters produced from the simulated annealing.

4.2 Write the modeled profile generated in section 4.1 to an ASCII file.

5. Write the final fitted parameters to the output trending data file. Sample MTFESTIMATE trending output is shown in Figure 7.

Year	DOY	Path	Row	Target	Band	Azimuth	Optics	AT Det	XT Det	Integ	Phase	Fit	GSD	Range	Velocity	Water	Asym	NS	SRad	Width	Offset	SRad	Width	Offset
2000	261	022	039	Causeway	1	0.071766	2.3186e+01	42.303	40.863	3.600	-3.9371e-01	0.449	30.000	705744.375	6839.098	74.8	0.0	2	103.0	10.00	-17.20	98.3	10.00	17.20
2000	261	022	039	Causeway	2	0.071861	2.3335e+01	42.303	40.863	3.600	-4.6986e-01	0.439	30.000	705742.438	6839.057	74.8	0.0	2	103.3	10.00	-17.20	98.1	10.00	17.20
2000	261	022	039	Causeway	3	0.071831	2.3185e+01	42.303	40.863	3.600	2.4909e-01	0.590	30.000	705750.875	6839.282	54.1	0.0	2	86.3	10.00	-17.20	83.2	10.00	17.20
2000	261	022	039	Causeway	4	0.071698	2.3520e+01	42.303	40.863	3.600	-5.6739e-01	0.691	30.000	705748.562	6839.179	42.7	0.1	2	87.6	10.00	-17.20	82.7	10.00	17.20
2000	261	022	039	Causeway	5	0.071737	2.3719e+01	42.303	40.863	3.600	-1.3014e+00	0.342	30.000	705746.312	6839.127	12.8	0.0	2	36.4	10.00	-17.20	33.2	10.00	17.20
2000	261	022	039	Causeway	6	0.072200	2.2190e+01	42.303	40.863	3.600	-1.4280e+00	0.970	30.000	705756.188	6839.589	12.2	0.0	2	59.9	10.00	-17.20	55.5	10.00	17.20
2000	261	022	039	Causeway	7	0.072177	2.2230e+01	42.303	40.863	3.600	-1.2534e+00	0.742	30.000	705753.875	6839.504	11.4	-0.0	2	47.8	10.00	-17.20	46.5	10.00	17.20
2000	261	022	039	Causeway	8	0.071943	1.4302e+01	21.152	20.431	0.660	1.8103e+00	0.587	15.000	705740.812	6839.059	22.8	-0.1	2	72.3	10.00	-17.20	66.9	10.00	17.20
2000	261	022	039	Causeway	9	0.072099	2.2244e+01	42.303	40.863	3.600	-1.5391e+00	0.953	30.000	705758.938	6839.658	12.2	0.0	2	60.0	10.00	-17.20	55.5	10.00	17.20
2000	261	022	039	I-10_Old	1	-0.516236	1.7729e+01	42.303	40.863	3.600	-4.3485e+00	0.644	30.000	705829.562	6842.794	70.5	-0.0	2	89.0	14.20	-15.15	88.7	14.20	15.15
2000	261	022	039	I-10_Old	2	-0.516559	1.7895e+01	42.303	40.863	3.600	-3.9627e+00	0.677	30.000	705827.688	6842.774	70.6	-0.0	2	88.6	14.20	-15.15	89.1	14.20	15.15
2000	261	022	039	I-10_Old	3	-0.514778	1.7695e+01	42.303	40.863	3.600	-3.9253e+00	0.756	30.000	705836.250	6843.705	48.2	-0.0	2	70.8	14.20	-15.15	70.6	14.20	15.15
2000	261	022	039	I-10_Old	4	-0.515239	1.8426e+01	42.303	40.863	3.600	-4.1640e+00	0.982	30.000	705834.375	6843.275	36.8	0.0	2	68.7	14.20	-15.15	68.7	14.20	15.15
2000	261	022	039	I-10_Old	5	-0.515680	1.8586e+01	42.303	40.863	3.600	-4.1405e+00	0.452	30.000	705831.438	6842.912	12.2	-0.1	2	28.8	14.20	-15.15	28.9	14.20	15.15
2000	261	022	039	I-10_Old	6	-0.515707	1.6931e+01	42.303	40.863	3.600	-5.3443e+00	1.332	30.000	705841.750	6842.901	12.0	-0.1	2	50.4	14.20	-15.15	46.3	14.20	15.15
2000	261	022	039	I-10_Old	7	-0.515193	1.6656e+01	42.303	40.863	3.600	-4.4182e+00	1.046	30.000	705839.125	6843.217	11.3	-0.1	2	41.9	14.20	-15.15	39.5	14.20	15.15
2000	261	022	039	I-10_Old	8	-0.516498	1.2675e+01	21.152	20.431	0.660	-1.8332e+00	0.671	15.000	705825.938	6842.743	20.5	-0.2	2	54.7	14.20	-15.15	56.1	14.20	15.15
2000	261	022	039	I-10_Old	9	-0.516339	1.6710e+01	42.303	40.863	3.600	-4.6544e+00	1.160	30.000	705844.438	6842.754	12.0	-0.1	2	49.5	14.20	-15.15	48.0	14.20	15.15
2001	273	044	034	San_Mateo	1	-0.975840	1.6700e+01	42.303	40.863	3.600	-2.2189e+00	0.672	30.000	706792.125	6830.640	65.2	-0.0	1	80.6	37.17	0.00			
2001	273	044	034	San_Mateo	2	-0.977612	1.6851e+01	42.303	40.863	3.600	-2.2453e+00	0.824	30.000	706794.812	6829.192	65.3	-0.0	1	80.7	37.17	0.00			
2001	273	044	034	San_Mateo	3	-0.978794	1.7252e+01	42.303	40.863	3.600	-2.2222e+00	0.739	30.000	706790.438	6830.167	47.2	-0.0	1	64.0	37.17	0.00			
2001	273	044	034	San_Mateo	4	-0.976428	1.8472e+01	42.303	40.863	3.600	-1.3846e+00	1.084	30.000	706789.938	6831.141	32.2	0.1	1	56.9	37.17	0.00			
2001	273	044	034	San_Mateo	5	-0.975340	1.9445e+01	42.303	40.863	3.600	-1.3551e+00	0.479	30.000	706790.500	6831.211	9.3	0.0	1	21.7	37.17	0.00			
2001	273	044	034	San_Mateo	6	-0.983885	1.6970e+01	42.303	40.863	3.600	-3.7628e-01	0.965	30.000	706795.938	6827.138	9.1	0.0	1	30.4	37.17	0.00			
2001	273	044	034	San_Mateo	7	-0.982368	1.7032e+01	42.303	40.863	3.600	9.3589e-01	0.843	30.000	706792.938	6828.102	8.9	0.0	1	27.1	37.17	0.00			
2001	273	044	034	San_Mateo	8	-0.979730	1.0666e+01	21.152	20.431	0.660	-5.7014e-02	1.420	15.000	706798.375	6827.743	18.0	0.3	1	43.5	37.17	0.00			
2001	273	044	034	San_Mateo	9	-0.984056	1.6984e+01	42.303	40.863	3.600	-5.0594e-01	1.025	30.000	706800.000	6827.118	9.1	0.1	1	30.3	37.17	0.00			
2011	290	120	035	Qingdao	5	-0.807173	2.2016e+01	42.303	40.863	3.600	5.2081e+00	0.352	30.000	706503.562	6843.522	11.6	-0.1	1	15.3	34.50	0.00			
2011	290	120	035	Qingdao	6	-0.812448	2.0777e+01	42.303	40.863	3.600	5.9862e+00	0.575	30.000	706510.438	6838.578	11.5	-0.1	1	19.1	34.50	0.00			
2011	290	120	035	Qingdao	7	-0.812091	1.9721e+01	42.303	40.863	3.600	6.3677e+00	0.574	30.000	706507.500	6838.932	10.9	-0.1	1	17.1	34.50	0.00			
2011	290	120	035	Qingdao	8	-0.805650	1.0351e+01	21.152	20.431	0.660	8.6866e+00	0.536	15.000	706508.188	6843.890	20.8	0.1	1	26.1	34.50	0.00			
2011	290	120	035	Qingdao	9	-0.812184	2.0495e+01	42.303	40.863	3.600	5.5529e+00	0.601	30.000	706514.188	6838.801	11.5	-0.1	1	19.0	34.50	0.00			

Figure 7: Sample MTFESTIMATE Trending Output Data

7.2.13.7.4 Phase 4. MTFPERFORM: Calculate OLI Spatial Performance

The fourth and final phase of the MTF characterization process operates on the trended output produced by MTFESTIMATE, for a collection of bridge targets observed at different orientation angles. It exploits the different STF cross-sections observed by the various targets to separate out the OLI cross-track and along-track optical blur parameters.

Recall that a single Gaussian optical blur parameter value is estimated and recorded for each target analyzed by MTFESTIMATE, and that this blur value (σ_m) and the associated target azimuth (θ_m) are recorded in the trending data record for each band/target. These are the Azimuth and Optics columns shown in the sample output table of Figure 7. Each of these optical blur observations are taken to be a combination of the underlying along- and across-track optical blur parameters, with the combination determined by the target azimuth. Thus, each target record provides an observation of the form:

$$\cos^2\theta_m \, \sigma^2_x + \sin^2\theta_m \, \sigma^2_a = \sigma^2_m$$

Where:

σ_x is the across-track Gaussian optical blur parameter
σ_a is the along-track Gaussian optical blur parameter

Once multiple targets at different orientation angles have been measured we have a series of observations of this form, from which we can solve for σ^2_x and σ^2_a. Note that we must also apply constraints so that:

$$\sigma^2_x >= 0 \text{ and } \sigma^2_a >= 0$$

making this a constrained least squares problem.

The unconstrained solution is:

$$\begin{bmatrix} \hat{\sigma}_x^2 \\ \hat{\sigma}_a^2 \end{bmatrix} = \begin{bmatrix} \sum_m \cos^2\theta_m \cos^2\theta_m & \sum_m \cos^2\theta_m \sin^2\theta_m \\ \sum_m \cos^2\theta_m \sin^2\theta_m & \sum_m \sin^2\theta_m \sin^2\theta_m \end{bmatrix}^{-1} \begin{bmatrix} \sum_m \cos^2\theta_m \, \sigma_m^2 \\ \sum_m \sin^2\theta_m \, \sigma_m^2 \end{bmatrix}$$

If the the unconstrained solution produces $\sigma^2_x >= 0$ and $\sigma^2_a >= 0$, then the unconstrained solution is the solution. Otherwise:

If $\sigma^2_x < 0$ then:

$$\begin{bmatrix} \hat{\sigma}_x^2 \\ \hat{\sigma}_a^2 \end{bmatrix} = \begin{bmatrix} 0 \\ \sum_m \sin^2\theta_m \, \sigma_m^2 \Big/ \sum_m \sin^2\theta_m \sin^2\theta_m \end{bmatrix}$$

If $\sigma^2_a < 0$ then:

$$\begin{bmatrix} \hat{\sigma}_x^2 \\ \hat{\sigma}_a^2 \end{bmatrix} = \begin{bmatrix} \sum_m \cos^2\theta_m \, \sigma_m^2 \Big/ \sum_m \cos^2\theta_m \cos^2\theta_m \\ 0 \end{bmatrix}$$

The σ_m and θ_m values contained in the input trending records for each band to be analyzed are used to calculate the summations shown in the equations above. These equations are then used to solve for σ^2_x and σ^2_a, from which we then calculate σ_x and σ_a. The results for each band are summarized in the output spatial performance report file, as shown in Figure 8.

Having solved for the across-track and along-track optical blur parameters, we use these values of σ_x and σ_a to construct the full across- and along-track transfer functions STF_x and STF_a:

Cross-Track: $STF_x(\omega) = O(\sigma_x, \omega)\, D(r_x, \omega)\, C(d_x, \omega)$
$STF_x(\omega) = \exp(-\omega^2 \sigma_x^2/2)\, \mathrm{sinc}(\omega\, r_x/2)\, \exp(-|\omega\, d_x|)$

Along-Track: $STF_a(\omega) = O(\sigma_a, \omega)\, D(r_a, \omega)\, I(\tau, \omega)\, C(d_a, \omega)$
$STF_a(\omega) = \exp(-\omega^2 \sigma_a^2/2)\, \mathrm{sinc}(\omega\, r_a/2)\, \mathrm{sinc}(\omega\, \tau\, V_g/2)\, \exp(-|\omega\, d_a|)$

These models are then used to evaluate the corresponding OLI spatial performance characteristics.

STF Fit Results for Band 1

Fit Based on 6 Trending Records:

Rec #	Year	DOY	WRS_Path	WRS_Row	Angle	Optics	Target
1	2000	261	022	039	0.071775	23.213	Causeway
2	2000	261	022	039	-0.515816	17.320	I-10_Old
3	2001	273	044	034	-0.975840	16.700	San_Mateo
4	2001	227	163	042	1.509833	20.330	King_Fahd_Causeway_W
5	2001	227	163	042	1.293609	19.314	King_Fahd_Causeway_C
6	2001	227	163	042	1.742302	21.356	King_Fahd_Causeway_E

OLI STF Fitted Parameters:
Cross-Track Optics: 20.524 urad Along-Track Optics: 19.430 urad
Cross-Track Detector: 40.863 urad Along-Track Detector: 42.303 urad
Charge Diffusion: 0.000 urad Integration Time: 3.600 msec

Figure 8: Sample MTFPERFORM Solution Summary

Parameter Error Estimates

The unconstrained least squares solution can also be used to estimate the errors in the σ^2_x and σ^2_a, parameters determined from the solution. First, the residuals (v_i) must be computed for each of the input trending records used in the solution:

$$v_i = \sigma^2_i - (\cos^2\theta_i\, \sigma^2_x + \sin^2\theta_i\, \sigma^2_a)$$

using the best-fit values of σ^2_x and σ^2_a to solve for the difference between the measured blur values (σ^2_i) and the corresponding modeled values ($\cos^2\theta_i\, \sigma^2_x + \sin^2\theta_i\, \sigma^2_a$). The residuals are then used to compute the variance of an observation of unit weight:

$$\sigma_{UW}^2 = \sum_{i=1}^{N} \mathrm{v_i}^2 \Big/ (N-2)$$

where N is the number of trending record observations used in the solution and (N-2) is the number of degrees of freedom in the solution (number of observations minus number of parameters). The estimated parameter covariance matrix is then:

$$Cov = \sigma_{UW}^2 \begin{bmatrix} \sum_m \cos^2\theta_m \cos^2\theta_m & \sum_m \cos^2\theta_m \sin^2\theta_m \\ \sum_m \cos^2\theta_m \sin^2\theta_m & \sum_m \sin^2\theta_m \sin^2\theta_m \end{bmatrix}^{-1}$$

The diagonal terms of this covariance matrix are the aposteriori parameter estimate variances. Note that since it is the optical blur variances (σ^2_x and σ^2_a) that were estimated, the covariance terms apply to the σ^2 values, not the σ values.

To add conservatism to the performance estimates used to evaluate the OLI Key Performance Requirement (KPR) relating to spatial response (KPR #4), we use the covariance matrix to compute the 90% confidence interval bounding the best-fit optical blur parameter estimates. We then use the 90% confidence interval limiting value that represents the best OLI spatial performance (i.e., smallest blur) that is consistent with the observed data to within the specified confidence. We do this by applying the 90% circular error scaling factor (2.146) to the estimate standard deviations, and subtracting the resulting value from the best fit parameter value:

$$\sigma_x^2(90\%) = MAX\left(\sigma_x^2 - 2.146 * \sqrt{Cov(1,1)}, 0\right)$$
$$\sigma_a^2(90\%) = MAX\left(\sigma_a^2 - 2.146 * \sqrt{Cov(2,2)}, 0\right)$$

The resulting 90% confidence values of $\sigma_x(90\%)$ and $\sigma_a(90\%)$ can then be used in the OLI STF model to compute conservative (best case) estimates for the OLI edge slope and edge extent parameters. Note that the fewer the available input records, the larger will be the difference between the best-fit results and the best case 90% confidence results.

Calculating OLI Spatial Performance Parameters

The along- and across-track OLI system transfer function models are evaluated using the profile sequence length (N) and oversampling parameter (O_s) values specified in the MTFPERFORM input ODL parameter file. Note that these values need not be the same as the values used during the model estimation procedure. A higher resolution (larger N and O_s values) model would typically be used at this stage to increase the fidelity of the edge response model used to determine edge slope and half edge extent.

The along- and across-track STF models, which are formulated in sensor units, are scaled to ground units using the nominal Landsat altitude (705 km). They are then inverse Fourier transformed to yield the corresponding along- and across-track line spread functions (LSFs). These are then integrated to compute the along- and across-track edge spread/response functions (ESFs):

ESF(0) = LSF(0)
for i = 1 to N-1

ESF(i) = ESF(i-1) + LSF(i)

Where:
ESF(i) = edge spread function at index i
LSF(i) = line spread function at index i

We next interpolate the X locations corresponding to the 5%, 40%, 50%, 60%, and 95% edge response points. Since, by construction, the ESF is normalized so that the minimum value is 0 and the maximum value is 1, this amounts to finding the ESF indicies that straddle 0.05, 0.4, 0.5, 0.6, and 0.95 and then linearly interpolating the exact value:

```
for v in {0.05, 0.4, 0.5, 0.6, 0.95}
    i = 1
    while i < N and ESF(i) < v
        i = i+1
    pos(v) = (i-1) + (v – ESF(i-1)) / (ESF(i) – ESF(i-1))
```

We then calculate the edge slope from the pos(0.4) and pos(0.6) values:
Edge Slope = O_s * (0.6 – 0.4) / (pos(0.6) – pos(0.4)) / GSD

We calculate the half edge extent from the pos(0.05), pos(0.5), and pos(0.95) values:
Low half edge extent = GSD / O_s * (pos(0.5) – pos(0.05))
High half edge extent = GSD / O_s * (pos(0.95) – pos(0.5))
Half Edge Extent = MAX(Low half edge extent, High half edge extent)

The edge slope and half edge extent values determined for each band are summarized at the end of the output spatial performance report file, as shown in Figure 9. Note that spatial performance estimates are computed using both the best-fit OLI STF parameter values and the best case 90% confidence interval values described above.

Edge Slope and Half Edge Extent Spatial Performance by Band

Band	NObs	XT_ES	XT_HEE	AT_ES	AT_HEE	ES_Spec	HEE_Spec
1	6	0.02338	27.451	0.02207	29.031	> 0.027	< 23.0
2	6	0.02317	27.709	0.02207	29.026	> 0.027	< 23.0
3	6	0.02301	27.924	0.02204	29.065	> 0.027	< 23.0
4	6	0.02273	28.287	0.02139	30.005	> 0.027	< 23.5
5	6	0.02245	28.652	0.02135	30.054	> 0.027	< 24.0
6	6	0.02379	26.953	0.02253	28.389	> 0.027	< 28.0
7	6	0.02399	26.708	0.02254	28.378	> 0.027	< 29.0
8	6	0.03813	16.955	0.03568	18.128	> 0.054	< 14.0
9	0	0.02399	26.708	0.02254	28.378	> 0.027	< 27.0

Best Case Edge Slope and Half Edge Extent Spatial Performance by Band

Band	NObs	XT_ES	XT_HEE	AT_ES	AT_HEE	ES_Spec	HEE_Spec
1	6	0.02540	25.112	0.02287	27.946	> 0.027	< 23.0
2	6	0.02516	25.370	0.02287	27.938	> 0.027	< 23.0
3	6	0.02480	25.775	0.02279	28.055	> 0.027	< 23.0
4	6	0.02432	26.318	0.02202	29.093	> 0.027	< 23.5
5	6	0.02384	26.890	0.02193	29.221	> 0.027	< 24.0

```
6        6      0.02562    24.878     0.02326    27.430     > 0.027      < 28.0
7        6      0.02580    24.687     0.02324    27.449     > 0.027      < 29.0
8        6      0.04271    15.089     0.03739    17.282     > 0.054      < 14.0
9        0      0.02580    24.687     0.02326    27.430     > 0.027      < 27.0
```

Figure 9: Sample Spatial Performance Summary Output

In the likely event that there are no trended output results for the cirrus band (band 9), the results for the other two short-wave infrared bands (band 6: SWIR1, and band 7: SWIR2) will be used to estimate cirrus performance. This is done solely for purposes of key performance requirement evaluation per the Key Performance Requirements Evaluation Plan (see below). To effect this, the cirrus band estimated edge slopes are taken to be the larger of the band 6 and band 7 results, whereas the cirrus band edge extents are taken to be the smaller of the band 6 and band 7 results:

Cirrus XT edge slope = MAX(SWIR1 XT edge slope, SWIR2 XT edge slope)
Cirrus AT edge slope = MAX(SWIR1 AT edge slope, SWIR2 AT edge slope)
Cirrus XT edge extent = MIN(SWIR1 XT edge extent, SWIR2 XT edge extent)
Cirrus AT edge extent = MIN(SWIR1 AT edge extent, SWIR2 AT edge extent)

7.2.13.8 Maturity

1. This ADD provides a completely reworked version of the algorithm.

Some additional background assumptions and notes include:

1. Bridge region of interest polygons are contained in the target definition file for each target WRS path/row. These polygons are designed to avoid undesirable areas such as bridge crossovers.
2. L0R file name is currently used to extract acquisition date.
3. Target model initialization parameters include; reflectance of bridge span(s) (assumed to be the same for all), reflectance of water, reflectance asymmetry, width of bridge span(s), bridge span offset(s), and GSD of bridge pixels.
4. System transfer function initialization parameters include; optical Gaussian component, XT detector size, AT detector size, integration time, exponential decay (=0), and phase.
5. Target simplex bounds include; reflectance of bridge span(s) (assumed to be the same for all), reflectance of water, reflectance asymmetry, width of bridge span(s), bridge span offset(s), and GSD of bridge pixels.
6. System transfer function simplex bounds include; optical Gaussian component, XT detector size, AT detector size, integration time, exponential decay (=0), and phase.
7. Implementation has the bridge and system models plus simplex bounds stored within a MTF systems file.
8. Contents of MTF bridge characterization report:
   a. Scene acquisition date and WRS path/row
   b. For each target and for each band;
   Geometric parameters: GSD, target range, velocity, and orientation
   Target parameters: target name, number of spans, span radiances, water radiance, water asymmetry, span widths, span offsets
   System transfer function parameters: optical Gaussian, AT detector size, XT detector size, integration time, exponential decay, phase.
   Fit statistics: Root mean squared error to model.

7.3 TIRS Geometry Algorithms

7.3.1 TIRS Line-of-Sight Model Creation

7.3.1.1 Background/Introduction

The line-of-sight (LOS) model creation algorithm gathers the ancillary data and calibration parameters required to support geometric processing of the input TIRS image data set; validates the image time codes; extracts, validates, and preprocesses the TIRS scene select mirror (SSM) telemetry contained in the ancillary data stream; extracts the corresponding ephemeris and attitude data from the ancillary data stream; performs the necessary coordinate transformations; and stores the results in a geometric model structure for subsequent use by other geometric algorithms. The TIRS LOS model creation algorithm is derived from the OLI model creation algorithm. Its implementation will be very similar to the corresponding OLI application and will draw on the same spacecraft model, math, and utility libraries. Note that the ephemeris and attitude preprocessing logic common to both sensors is performed by the ancillary data preprocessing algorithm to isolate the bulk of the geometric processing logic from the details of the incoming ancillary data stream. New attitude data processing logic has also been added to separate the high- and low-frequency attitude effects to allow the image resampling process to better correct for jitter at frequencies above the original 10 Hz algorithm design limit without requiring an unreasonably dense resampling grid.

7.3.1.2 Dependencies

The TIRS LOS Model Creation algorithm assumes that the Ancillary Data Preprocessing algorithm has been executed to accomplish the following:

Validated ephemeris data for the full imaging interval have been generated
Validated attitude data for the full imaging interval have been generated
The ancillary data have been scaled to engineering units

The Ancillary Data Preprocessing algorithm will generate preprocessed smoothed and cleaned ephemeris and attitude data streams. This provides a standard validated input for subsequent LOS model generation.

7.3.1.3 Inputs

The TIRS LOS Model Creation algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data; including data set IDs to provide unique identifiers for data trending).

Algorithm Inputs
ODL File (implementation)
Acquisition Type (Earth, Lunar, Stellar) (optional, defaults to Earth)
CPF File Name
Preprocessed Ancillary Data Input File Name
L0R/L1R File Name
WRS Path/Row (stored in model and used for trending)
Trending On/Off Switch
Geometric work order common characterization ID (for trending)
Work Order ID (for trending)
Optional Precision Model Input Parameters (see note 6)
Input Precision Model Reference Time (optional)
Input Precision Ephemeris Correction Order (optional)

Input Precision X Correction Parameters (optional)
Input Precision Y Correction Parameters (optional)
Input Precision Z Correction Parameters (optional)
Input Precision Attitude Correction Order (optional)
Input Precision Roll Correction Parameters (optional)
Input Precision Pitch Correction Parameters (optional)
Input Precision Yaw Correction Parameters (optional)
CPF Contents
WGS84 Earth ellipsoid parameters
Earth orientation parameters (UT1UTC, pole wander, leap seconds) (see note 1)
Earth rotation velocity
Speed of light
TIRS to ACS reference alignment matrix/quaternion
Spacecraft center of mass (CM) to TIRS offset in ACS reference frame (new)
High frequency attitude data cutoff frequency (Hz)
Scene select mirror calibration parameters (new) (see Table 1 below)
Focal plane model parameters (Legendre coefficients)
TIRS detector delay table (including whole pixel deselect offsets) (see note 10)
Nominal L0R fill (per band/SCA)
Nominal TIRS frame time nominal_frame_time (14.2857143 msec)
Nominal TIRS integration time
Image time code outlier thresholds delta_time_tolerance (DTIME_TOL) and time_outlier_tolerance (OUTLIER_TOL) (see note 3)
SSM encoder outlier threshold (see note 7)
Preprocessed Ancillary Data Contents
Attitude Data
Attitude data UTC epoch: Year, Day of Year, Seconds of Day
Time from epoch (one per sample, nominally 50 Hz)
ECI quaternion (vector: q1, q2, q3, scalar: q4) (one per sample)
ECEF quaternion (one per sample)
Body rate estimate (roll, pitch, yaw rate) (one per sample)
Roll, pitch, yaw estimate (one per sample)
Ephemeris Data
Ephemeris data UTC epoch: Year, Day of Year, Seconds of Day
Time from epoch (one per sample, nominally 1 Hz)
ECI position estimate (X, Y, Z) (one set per sample)
ECI velocity estimate (Vx, Vy, Vz) (one set per sample)
ECEF position estimate (X, Y, Z) (one set per sample)
ECEF velocity estimate (Vx, Vy, Vz) (one set per sample)
L0R/L1R Data Contents
Image Time Codes (one per line)
Integration Time
Scene Select Mirror Telemetry Packets (from Ancillary Data, see Table 2 below)
TIRS Ancillary Data Time Code (one per 1 Hz frame)
Mirror Encoder Readout (24 bits, in counts) (20 samples per 1 Hz frame) (see note 9)
Detector Alignment Fill Table (see note 2)

7.3.1.4 Outputs

TIRS LOS Model (additional detail is provided in Table 3 below)
WGS84 Earth ellipsoid parameters
Earth Orientation Parameters (for current day) from CPF
Earth rotation velocity

Speed of light
TIRS to ACS reference alignment matrix/quaternion
Spacecraft center of mass to TIRS offset in ACS reference frame
SSM model parameters (Telescope alignment matrix and preprocessed SSM angles)
Focal plane model parameters (Legendre coefs)
Detector delay table (including whole pixel deselect offsets)
Nominal detector alignment fill table (from CPF)
L0R detector alignment fill table (from L0R)
ECI J2000 spacecraft ephemeris model (original and corrected)
ECEF spacecraft ephemeris model (original and corrected)
Spacecraft attitude model (time, roll, pitch, yaw) (orig and corr) (see note 4)
High frequency attitude perturbations (roll, pitch, yaw) per image line (jitter table)
Image time codes (see note 5) (in seconds)
Integration Time (in seconds)
Sample Time (in seconds)
WRS Path/Row
Model Trending Data
WRS Path/Row
Acquistion Date/Time
Geometric work order common characterization ID
Work Order ID
Image start UTC time (year, day of year, seconds of day)
Computed image frame time (in seconds)
Number of image lines
Number of out of limit image time codes
Number of out of limit SSM time codes
Number of out of limit SSM encoder measurements

7.3.1.5 Options

Trending On/Off Switch
Optional precision model input parameters can be used to force model corrections.

7.3.1.6 Prototype Code

Input to the executable is an ODL file; output is a HDF4 formatted TIRS model file.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The following text is a brief description of the main set of modules used within the prototype with each module listed along with a very short description. It should be noted that not all library modules are referenced in the explanations below. The modules within the main create directory of the prototype are discussed and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters. Note that the modules in the main “create” directory are the same as the corresponding OLI routines. The TIRS-specific differences reside in the library routines.

model_create – Main procedure that retrieves the input parameters and invokes the model generation and model output logic.

getpar – Retreives the user-provided ODL parameters.

oli_zero_model – Library routine that initializes the model structure. Adapted from the corresponding OLI routine. Note that since many of the geometric model routines involve substantial reuse of heritage OLI code, many have not been renamed. This made it possible to leave the model creation driver routines unchanged.

get_path_row_l0ra - Designed to retrieve the WRS path and row numbers from the L0R data. In the baseline algorithm these are ODL input parameters but they should ultimately be extracted from the Level 0R data directly. This unit is a placeholder for the time being.

oli_run_model – Library routine that loads the CPF, L0R, and preprocessed ancillary data into the model structure.

oli_get_cpf – Library routine that reads the CPF. Largely reuse from the corresponding OLI CPF input routine with new TIRS SCENE_SELECT_MIRROR group added.

tirs_get_scene_select_mirror – Library routine that reads the new TIRS SCENE_SELECT_MIRROR calibration parameter group from the CPF.

oli_get_model_sensor_params – Library routine that loads the sensor section of the model structure using data from the CPF and the L0R frame header. Adapted from corresponding OLI routine.

oli_get_model_image_params – Library routine that loads the image section of the model structure using data from the CPF, the L0R line headers, and the L0R detector offset fields. This unit also validates the image line time codes. Adapted from corresponding OLI routine.

tirs_get_ssm_from_l0r – Library routine that reads and preprocesses the TIRS scene select mirror (SSM) telemetry extracted from the L0R data, and loads it into the TIRS geometric model.

tirs_align_ssm_data – Library routine that aligns the TIRS SSM encoder telemetry samples with the 1 Hz ancillary data frames, so that there are 20 complete samples per frame.

tirs_check_ssm_data – Library routine that quality checks the TIRS SSM telemetry time codes and encoder angle data.

tirs_smooth_ssm_data – Library routine that applies a moving window smoother to the TIRS SSM encoder data.

oli_get_model_earth_params – Library routine that loads the Earth model parameters from the CPF. Reuse of OLI routine.

oli_get_ancillary_pre – Library routine that loads the attitude and ephemeris model sections using data from the preprocessed ancillary data file. Reuse of OLI routine.

oli_build_jitter_table – Library routine that splits the attitude data from the ancillary data into low- and high-frequency streams, interpolates the high frequency data to match the TIRS line

times, stores this per image line high frequency attitude data in the jitter table structure, and replaces the original combined attitude data stream with the low-frequency stream.

remez – Library routine that uses the Remez exchange algorithm to synthesize the weights (taps) of a low pass finite impulse response digital filter based on input filter size and cutoff frequency parameters. GNU Public License code written by Jake Janovetz, formerly of UIUC, which is available online at his site: http://www.janovetz.com/jake/ and more specifically:
http://www.janovetz.com/jake/remez/remez-19980711.zip

l8_correct_attitude – Library routine that applies the user-input precision model attitude corrections (if any).

l8_convert_ephem – Library routine that applies the user-input precision model ephemeris corrections (if any).

oli_put_model – Library routine that writes the TIRS model structure to the output HDF model file. Adapted from corresponding OLI routine.

7.3.1.7 Procedure

The TIRS LOS model is stored as a structure and is created from information contained in the Level 0R or Level 1R image data, the CPF, and the Ancillary data. The model is subsequently used along with the CPF to create a resampling grid. Data present in the model structure includes satellite position, velocity, and attitude, line-of-sight (LOS) angles, timing references, scene select mirror position, precision correction information (if any), and the software version. The TIRS LOS model is also used in several characterization and calibration routines for mapping input line/sample locations to geographic latitude/longitude.

The TIRS LOS model may be thought of in two parts, an instrument model that provides a line-of-sight vector for each TIRS detector (and, hence, each image line/sample), and a spacecraft model that provides spacecraft ephemeris (position and velocity) and attitude as a function of time. These models are linked by the image time stamps that allow each Level 0R or Level 1R image sample to be associated with a time of observation. The spacecraft portion of the model is common to the OLI LOS model.

Instrument Model
The arrangement of the bands and SCAs on the TIRS focal plane is shown in Figure 1. The model treats every band of every SCA independently. This is done by defining a set of 3rd order Legendre polynomials (see maturity note #2) for each band of each SCA. Unlike the OLI, the TIRS detectors are arranged in a two-dimensional array with two rows of that array being downlinked for each spectral band. One of the downlinked rows is primary and the other, redundant row is only used to replace bad pixels in the primary row. The TIRS LOS Legendre polynomials represent a theoretical “nominal” set of detectors that are best-fit to the primary row of detectors. This approach treats any replaced detectors as though they were aligned with the primary detectors for purposes of sensor LOS generation. This is a simplification of the OLI approach which also must account for even/odd detector stagger. In the TIRS case, this stagger is effectively set to zero. This approach explicitly models any offsets caused by detector replacement, and the sub-pixel deviations of each detector from its nominal (Legendre best fit) location, for correction during image resampling. This leads to three detector types: nominal, actual, and exact. A nominal detector is calculated from the Legendre

polynomials. An actual detector corrects the nominal detector location for the nominal (whole pixel) pixel select offsets. Like the OLI, since individual detectors may be deselected/replaced, these offsets are detector dependent. An exact detector has the actual correction applied but also includes the specific individual (sub-pixel) detector offsets. The Legendre polynomials and a table of detector offset values are stored in the CPF.

There is a slight angular difference between the line of sight vectors or angles associated with the primary and any replaced detectors. If the nominal LOS, generated using the 3rd order Legendre model, is $\psi_{nominal}$, the look angles for the actual and exact detectors are:

$$\psi_{x\_actual} = \psi_{x\_nominal} + \text{round}(\text{detector\_shift\_x}) * \text{IFOV}$$
$$\psi_{y\_actual} = \psi_{y\_nominal} + \text{round}(\text{detector\_shift\_y}) * \text{IFOV}$$

$$\psi_{x\_exact} = \psi_{x\_nominal} + \text{detector\_shift\_x} * \text{IFOV}$$
$$\psi_{y\_exact} = \psi_{y\_nominal} + \text{detector\_shift\_y} * \text{IFOV}$$

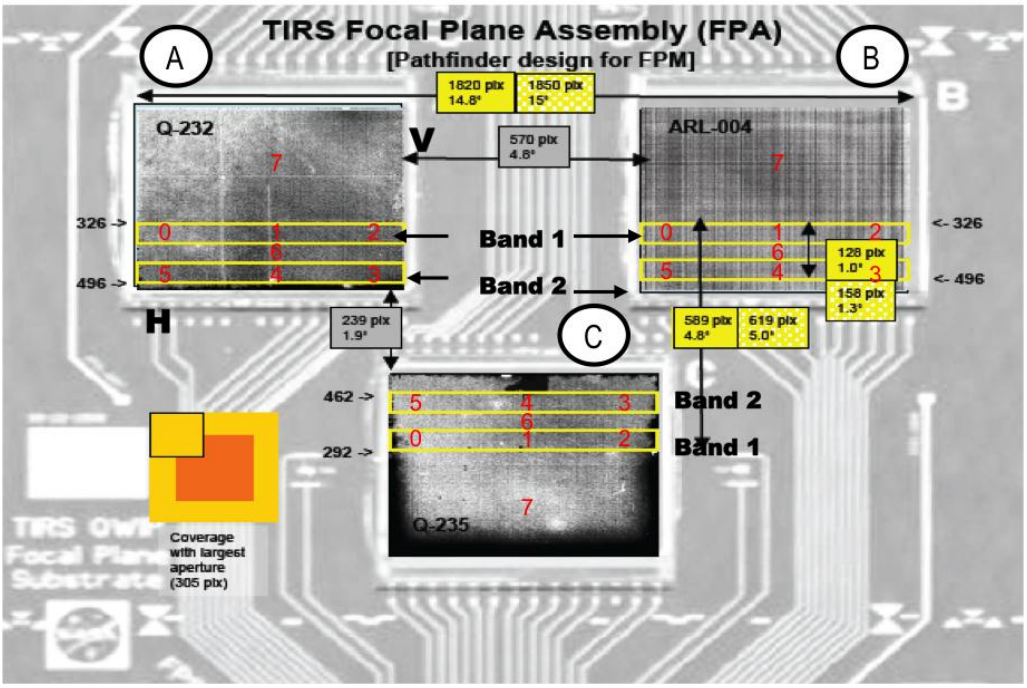


Figure 1: TIRS Focal Plane Layout

The detector_shift_x and detector_shift_y values are the detector-specific offsets stored in the CPF detector delay tables. These offsets include both the whole-pixel deselect/replacement offsets and the fractional-pixel detector placement effects, and must be rounded to extract the integer portion. Note that the integer portion of the detector_shift_y value is always zero since the deselect effects are applicable only in the X direction. Also note that the integer portion of the detector_shift_x values will also all be zero in the event that no primary detectors are bad.

The nominal LOS is used in most line-of-sight projection applications. The actual LOS is used in conjunction with the actual image time (see below) to model the errors introduced by trading time (sample delay) for space (detector offset) for purposes of correcting the nominal LOS model. The exact LOS is generally used only for data simulation and other analytical purposes rather than in the geometric correction model, as the sub-pixel portion of the detector delay is applied directly in the image resampler rather than being included in the LOS model.

Sample Timing

The TIRS provides a time stamp with each image line collected. These time stamps make it possible to relate the image samples (pixels) to the corresponding spacecraft ephemeris and attitude data. The TIRS time stamps are contained in a line header that precedes each image line. The TIRS line header contents are shown in Figure 2. Several items in this figure are worthy of particular note. First, the time stamp associated with a data frame is recorded at the beginning of the detector integration period. Second, the line header includes the integration time (TIRS does not use a separate frame header) and identifies the detector rows selected for downlink for each band and each SCA. This includes the dark band which is not included in the TIRS geometric model. The line header fields other than the time code should be static within an imaging interval.

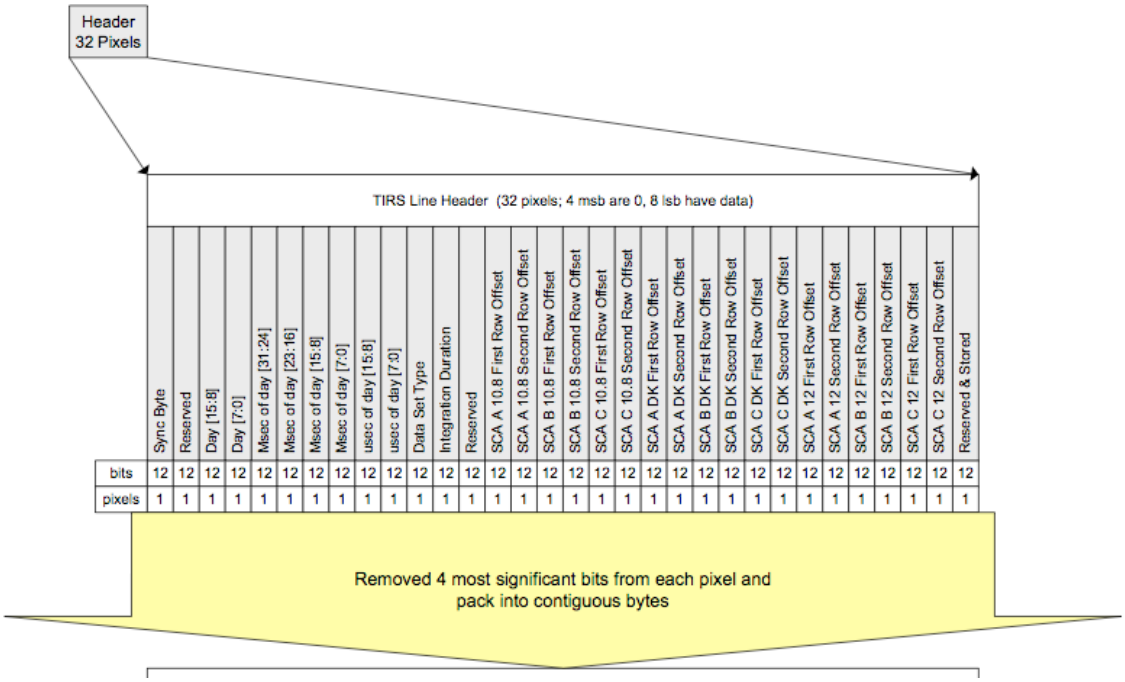


Figure 2: TIRS Line Header Contents

Note that having the time code define the start of detector integration is different than the OLI where the time code represents the end of integration. This has the effect of making the integration time correction a positive adjustment to the pixel time for TIRS rather than a negative adjustment, as is the case for the OLI.

Also note that the TIRS line header is composed of 12-bit data words, like the other TIRS “pixels”, where the most significant 4 bits are all zero. The assumption here is that Level 0 processing will treat the line header words the same way that it handles TIRS pixel data and repackage the 12-bit fields into 16-bit data words. If instead, Level 0 processing strips off the extra 4 bits from the line header fields, then the line header preprocessing step mentioned below, will not be necessary.

One further complication to the problem of assigning times to image samples is the fact that the Level 0R/1R input imagery may include fill pixels inserted to achieve nominal primary and replaced detector alignment. This fill insertion allows the geometrically unprocessed 0R/1R imagery to be viewed as a spatially contiguous image without detector misalignments. The amount of detector alignment fill present will be indicated in the L0R/L1R image data (this is the purpose of the detector alignment fill table input noted above) so that the association of image samples with their corresponding time

stamps can be adjusted accordingly. The assumption here is that, like OLI data, image fill will not be used to achieve nominal SCA or band alignment for TIRS data.

Due to the potential for deselected/replaced detectors, the nominal and actual times associated with a given pixel may not be the same. The actual time reflects the time that the current detector was actually sampled whereas the nominal time reflects the time at which the idealized detector represented by the TIRS LOS model would have been sampled.

If the current position within the image is given as a line and sample location, the two different "types" of times for TIRS pixels are calculated by:

```
l0r_fill_pixels = round(detector_shift_x) + nominal_fill_pixels
time_index = line_number - l0r_fill_pixels
if ( time_index < 0 ) time_index = 0
if (time_index > (num_time_stamps - 1)) time_index = num_time_stamps - 1

actual_time = line_time_stamp[time_index] + integration_time/2
                    + (line_number - l0r_fill_pixels - time_index) * TIRS_sample_time

nominal_time = actual_time + round(detector_shift_x) * TIRS_sample_time
```

where:

- line_number is the zero-referenced TIRS image line number (N).
- nominal_fill_pixels is the amount of detector alignment fill to be inserted at the beginning of pixel columns that correspond to nominal detectors; that is, those detectors with a delay value of zero that are the basis for the Legendre polynomial LOS model. This value comes from the CPF and will be zero if there are no bad detectors to replace.
- l0r_fill_pixels is the total amount of detector alignment fill to be inserted at the beginning of the pixel column associated with the current detector. It includes both the nominal_fill_pixels and the detector-specific delay fill required to align deselected/replaced detectors.
- num_time_stamps is the total number of time codes (image lines) in the image. It is tested to ensure that time_index, the line_time_stamp index, does not go out of bounds.
- detector_shift_x is the amount of detector offset for the current detector from the TIRS LOS model detector delay table. It is rounded to the nearest integer pixel because time offsets can only occur in whole line increments.

The detector_shift_x parameter is the detector-specific along-track offset as recorded in the CPF and subsequently stored in the LOS model detector delay table. It is rounded to the nearest integer so as to include the effects of even/odd detector stagger and detector deselect but not the detector-specific sub-pixel offsets. The L0R/L1R data can be accessed by SCA making the association of sample number with detector index more straightforward. Note that, like OLI, the TIRS Level 0R data organizes the image samples from all 3 TIRS SCAs so that the samples are numbered left-to-right for all SCAs. This convention is also followed in the CPF detector offset tables. There are 640 samples per SCA for each spectral band.

Note that when fill is used to align replaced detectors the spatial difference between the nominal and actual look vectors is approximately compensated by the time difference between $t_{nominal}$ and t_{actual}.

TIRS Scene Select Mirror Model

The TIRS scene select mirror (SSM) redirects the lines-of-sight from the TIRS telescope, which is oriented with its optical axis nominally in the +X direction, toward either: 1) the nadir Earth view; or 2) the space view port; or 3) the internal black body. It is the Earth view case that is of interest to the geometric processing models. Figure 3 shows the SSM and TIRS telescope in relationship to the TIRS coordinate system, in which the telescope optical axis defines the +X axis. The TIRS coordinate system is nominally aligned with the spacecraft coordinate system (+X toward the direction of flight, +Z toward nadir, +Y completing a right-handed coordinate system).

The SSM angle, γ, is defined as the angle between the SSM normal and the SSM axis of rotation. This angle is nominally $\pi/4$ radians (45 degrees). The SSM angle is a parameter of the SSM system that would be stored in the Calibration Parameter File (CPF).

Define a scene select mirror coordinate system, nominally parallel to the TIRS coordinate system, with the +X axis parallel to the SSM axis of rotation ($\underline{\mathbf{X}}_{ssm}$), the +Y axis in the direction of the cross product of the mirror normal vector $\underline{\mathbf{n}}_{ssm}$ and $\underline{\mathbf{X}}_{ssm}$, and the +Z axis completing a right handed coordinate system.

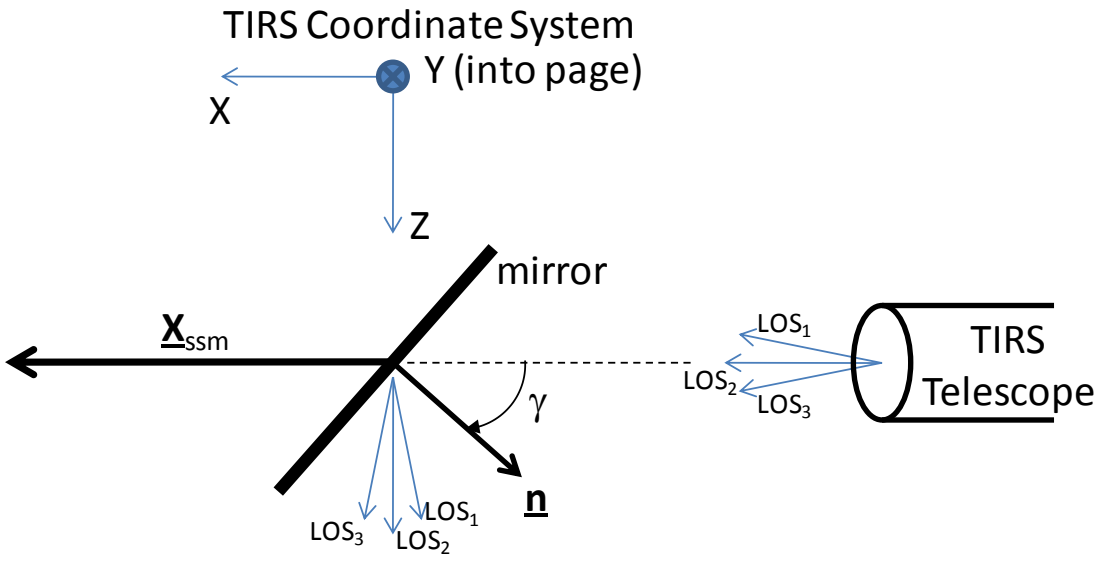
The mirror axis of rotation and mirror normal are:

$$\underline{\mathbf{X}}_{ssm} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \underline{\mathbf{n}}_{ssm} = \begin{bmatrix} -\cos(\gamma) \\ 0 \\ \sin(\gamma) \end{bmatrix}$$

To include the effect of SSM rotation about its axis, rotate $\underline{\mathbf{n}}_{ssm}$ about $\underline{\mathbf{X}}_{ssm}$ by an angle $(\theta - \theta_0)$ where θ_0 is the SSM encoder angle at the nominal nadir pointing angle and θ is the actual SSM encoder angle reported in the TIRS ancillary data. The mirror normal as a function of θ is:

$$\underline{\mathbf{n}}_{ssm}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta - \theta_0) & -\sin(\theta - \theta_0) \\ 0 & \sin(\theta - \theta_0) & \cos(\theta - \theta_0) \end{bmatrix} \begin{bmatrix} -\cos(\gamma) \\ 0 \\ \sin(\gamma) \end{bmatrix} = \begin{bmatrix} -\cos(\gamma) \\ -\sin(\gamma)\sin(\theta - \theta_0) \\ \sin(\gamma)\cos(\theta - \theta_0) \end{bmatrix}$$

The nominal nadir pointing angle θ_0 would be a SSM calibration parameter stored in the CPF. The measured encoder angle as a function of time, $\theta(t)$, would be reported in the 1 Hz TIRS ancillary data stream, with 20 samples provided in each 1 Hz TIRS ancillary data packet. Any time delay between the actual encoder sample time(s) and the corresponding ancillary data packet time code would have to be accounted for, so this time delay, Δt, is included as a parameter in the CPF.



$\underline{\mathbf{X}}_{ssm}$ = Scene select mirror axis of rotation
$\underline{\mathbf{n}}$ = SSM normal vector
γ = mirror angle (nominally $\pi/4$)

Figure 3: Scene Select Mirror Line-of-Sight Redirection

In the TIRS telescope coordinate system, the LOS emerging from the telescope, $\underline{\mathbf{l}}_{tele}$, is:

$$\underline{\mathbf{l}}_{tele} = \begin{bmatrix} 1 \\ \tan(\delta_{XT}) \\ \tan(\delta_{AT}) \end{bmatrix}$$

where: δ_{XT} is the across-track LOS angle (from the Legendre polynomial model)
δ_{AT} is the along-track LOS angle (from the Legendre polynomial model)

To account for misalignments between the SSM and the TIRS telescope, rotate $\underline{\mathbf{l}}_{tele}$ about the X axis by an angle Δr, about the Y axis by an angle Δp, and about the Z axis by an angle Δy:

$$\mathbf{l}_{SSM} = \mathbf{M}(\Delta r, \Delta p, \Delta y)\,\mathbf{l}_{tele}$$

Where:

$$\mathbf{M}(\Delta r, \Delta p, \Delta y) = \begin{bmatrix} \cos\Delta y & \sin\Delta y & 0 \\ -\sin\Delta y & \cos\Delta y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Delta p & 0 & -\sin\Delta p \\ 0 & 1 & 0 \\ \sin\Delta p & 0 & \cos\Delta p \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\Delta r & \sin\Delta r \\ 0 & -\sin\Delta r & \cos\Delta r \end{bmatrix}$$

$$= \begin{bmatrix} \cos\Delta p\cos\Delta y & \sin\Delta r\sin\Delta p\cos\Delta y + \cos\Delta r\sin\Delta y & \sin\Delta r\sin\Delta y - \cos\Delta r\sin\Delta p\cos\Delta y \\ -\cos\Delta p\sin\Delta y & \cos\Delta r\cos\Delta y - \sin\Delta r\sin\Delta p\sin\Delta y & \cos\Delta r\sin\Delta p\sin\Delta y + \sin\Delta r\cos\Delta y \\ \sin\Delta p & -\sin\Delta r\cos\Delta p & \cos\Delta r\cos\Delta p \end{bmatrix}$$

The telescope misalignment angles, Δr, Δp, and Δy, are calibration parameters stored in the CPF.

The LOS vector is reflected off the SSM by multiplying it by the matrix **P**(θ):

$$\mathbf{P}(\theta) = \mathbf{I} - 2\,\underline{\mathbf{n}}_{SSM}(\theta)\,\underline{\mathbf{n}}^{T}_{SSM}(\theta)$$

And the resulting reflected LOS is:

$$\underline{\mathbf{l}}_{TIRS}(\theta) = \mathbf{P}(\theta)\,\underline{\mathbf{l}}_{SSM} = [\mathbf{I} - 2\,\underline{\mathbf{n}}_{SSM}(\theta)\,\underline{\mathbf{n}}^{T}_{SSM}(\theta)]\,\mathbf{M}(\Delta r,\Delta p,\Delta y)\,\underline{\mathbf{l}}_{tele}$$

Define $\hat{\theta} = \theta - \theta_0$, and $\gamma = \pi/4 + \Delta\gamma$, where $\Delta\gamma$ is the departure from the ideal mirror angle.

The corresponding reflection matrix, **P**, becomes:

$$\mathbf{P}\left(\hat{\theta},\Delta\gamma\right) = \begin{bmatrix} \sin 2\Delta\gamma & -\cos 2\Delta\gamma \sin\hat{\theta} & \cos 2\Delta\gamma \cos\hat{\theta} \\ -\cos 2\Delta\gamma \sin\hat{\theta} & \cos^2\hat{\theta} - \sin 2\Delta\gamma \sin^2\hat{\theta} & \sin\hat{\theta}\cos\hat{\theta}\left(1+\sin 2\Delta\gamma\right) \\ \cos 2\Delta\gamma \cos\hat{\theta} & \sin\hat{\theta}\cos\hat{\theta}\left(1+\sin 2\Delta\gamma\right) & \sin^2\hat{\theta} - \sin 2\Delta\gamma \cos^2\hat{\theta} \end{bmatrix}$$

For an ideal SSM, $\Delta\gamma$ = 0, so the ideal reflection matrix, $\mathbf{P}_0(\hat{\theta})$, becomes:

$$\mathbf{P}_0(\hat{\theta}) = \begin{bmatrix} 0 & -sin\hat{\theta} & cos\hat{\theta} \\ -sin\hat{\theta} & cos^2\hat{\theta} & sin\hat{\theta}\,cos\hat{\theta} \\ cos\hat{\theta} & sin\hat{\theta}\,cos\hat{\theta} & sin^2\hat{\theta} \end{bmatrix}$$

Which for nadir pointing ($\hat{\theta}$=0) reduces to:

$$\mathbf{P}_0(0) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

For a perfectly aligned SSM, Δr = Δp = Δy = 0, so:

$$\underline{\mathbf{l}}_{TIRS}(0) = \mathbf{P}_0(0)\,\underline{\mathbf{l}}_{tele} = \begin{bmatrix} \tan(\delta_{AT}) \\ \tan(\delta_{XT}) \\ 1 \end{bmatrix}$$

Note that this matches the nadir-pointing LOS formulation used for the OLI. To minimize the differences between the standard nadir-pointing (OLI) LOS model and the SSM reflected (TIRS) LOS model, we formulate the SSM effect as a rotation applied to a nadir-pointing LOS vector as follows:

Noting that: $\mathbf{P}_0(0)\,\mathbf{P}_0(0) = \mathbf{I}$

$$\underline{\mathbf{l}}_{TIRS}(\theta) = [\mathbf{I} - 2\,\underline{\mathbf{n}}_{SSM}(\theta)\,\underline{\mathbf{n}}^{T}_{SSM}(\theta)]\,\mathbf{P}_0(0)\,\mathbf{P}_0(0)\,\mathbf{M}(\Delta r,\Delta p,\Delta y)\,\mathbf{P}_0(0)\,\mathbf{P}_0(0)\,\underline{\mathbf{l}}_{tele}$$

$$\underline{\mathbf{l}}_{TIRS}(\theta) = [\mathbf{P}_0(\hat{\theta},\Delta\gamma)\,\mathbf{P}_0(0)]\,[\mathbf{P}_0(0)\,\mathbf{M}(\Delta r,\Delta p,\Delta y)\,\mathbf{P}_0(0)]\,\underline{\mathbf{l}}_{TIRS}(0)$$

The TIRS LOS, $\underline{\mathbf{l}}_{TIRS}(\theta)$, can thus be written as the ideal nadir-pointed LOS , $\underline{\mathbf{l}}_{TIRS}(0)$, rotated by the telescope alignment matrix, **M'**(Δr,Δp,Δy) and reflected by the SSM matrix, **P'**($\hat{\theta}$,$\Delta\gamma$):

$$\underline{\mathbf{l}}_{TIRS}(\theta) = \mathbf{P'}(\hat{\theta},\Delta\gamma)\ \mathbf{M'}(\Delta r,\Delta p,\Delta y)\ \underline{\mathbf{l}}_{TIRS}(0)$$

where:

$$\mathbf{P}'\left(\hat{\theta},\Delta\gamma\right)=\begin{bmatrix} \cos 2\Delta\gamma\cos\hat{\theta} & -\cos 2\Delta\gamma\sin\hat{\theta} & \sin 2\Delta\gamma \\ \sin\hat{\theta}\cos\hat{\theta}\left(1+\sin 2\Delta\gamma\right) & \cos^2\hat{\theta}-\sin 2\Delta\gamma\sin^2\hat{\theta} & -\cos 2\Delta\gamma\sin\hat{\theta} \\ \sin^2\hat{\theta}-\sin 2\Delta\gamma\cos^2\hat{\theta} & \sin\hat{\theta}\cos\hat{\theta}\left(1+\sin 2\Delta\gamma\right) & \cos 2\Delta\gamma\cos\hat{\theta} \end{bmatrix}$$

$$\mathbf{M}'\left(\Delta r,\Delta p,\Delta y\right)=$$

$$\begin{bmatrix} \cos\Delta r\cos\Delta p & -\sin\Delta r\cos\Delta p & \sin\Delta p \\ \cos\Delta r\sin\Delta p\sin\Delta y+\sin\Delta r\cos\Delta y & \cos\Delta r\cos\Delta y-\sin\Delta r\sin\Delta p\sin\Delta y & -\cos\Delta p\sin\Delta y \\ \sin\Delta r\sin\Delta y-\cos\Delta r\sin\Delta p\cos\Delta y & \sin\Delta r\sin\Delta p\cos\Delta y+\cos\Delta r\sin\Delta y & \cos\Delta p\cos\Delta y \end{bmatrix}$$

For an ideal SSM, the product of the reflection and alignment rotation matrices, $\mathbf{P'}_0(\hat{\theta},0)\ \mathbf{M'}(0,0,0)$, reduces to:

$$\mathbf{M}_0(\hat{\theta}) = \mathbf{P'}_0(\hat{\theta},0)\ \mathbf{M'}(0,0,0) = \begin{bmatrix} cos\hat{\theta} & -sin\hat{\theta} & 0 \\ sin\hat{\theta}\ cos\hat{\theta} & cos^2\hat{\theta} & -sin\hat{\theta} \\ sin^2\hat{\theta} & sin\hat{\theta}\ cos\hat{\theta} & cos\hat{\theta} \end{bmatrix}, \text{ and } \mathbf{M}_0(0) = \mathbf{I}$$

Noting that $\hat{\theta}$, $\Delta\gamma$, Δr, Δp, and Δy are all close to zero, it can be shown that the $\Delta\gamma$ mirror angle offset is approximately equivalent to a pitch misalignment of $2\Delta\gamma$. Using this approximation we can write the TIRS LOS transformation equation as the product of a reflection matrix, that is a function of only the mirror rotation angle θ, and a static telescope alignment matrix:

$$\underline{\mathbf{l}}_{TIRS}(\theta) \approx \mathbf{M}_0(\hat{\theta})\ \mathbf{M'}(\Delta r,\Delta p+2\Delta\gamma,\Delta y)\ \underline{\mathbf{l}}_{TIRS}(0)$$

In practice, the SSM reflection matrix $\mathbf{M}_0(\hat{\theta})$ will be close to **I** so it will be difficult to distinguish telescope misalignments, modeled by **M'**, from TIRS instrument misalignments, and corrections to any prelaunch telescope alignment parameters will be absorbed by the TIRS alignment angles estimated on-orbit. For this reason we do not anticipate performing on-orbit calibration for the SSM parameters. The TIRS SSM calibration parameters are summarized in Table 1. Note that since the primary (side A) and redundant (side B) SSM encoders are not perfectly aligned, the encoder values that correspond to nadir pointing will not be exactly the same. Thus, the nominal nadir encoder angle, θ_0, will be equal to either θ_A or θ_B depending on the mirror side/encoder in use. The mirror side in use will be indicated in the TIRS ancillary data.

Parameter	**Symbol**	**Nominal Value**	**Source**
Mirror Angle	γ	π/4 radians	Fixed constant
Mirror Angle Deviation	$\Delta\gamma$	0	Prelaunch characterization
Nadir Pointing Angle / Encoder Origin (Side A)	θ_A	0	Defined value – establishes reference for TIRS alignment calibration

Nadir Pointing Angle / Encoder Origin (Side B)	θ_B	0	Defined value – establishes reference for TIRS alignment calibration
Telescope Roll Offset	Δr	0	Prelaunch characterization
Telescope Pitch Offset	Δp	0	Prelaunch characterization
Telescope Yaw Offset	Δy	0	Prelaunch characterization
Encoder Time Offset	Δt	0	Prelaunch characterization

Table 1: TIRS Scene Select Mirror Model Calibration Parameters

Spacecraft Model
The spacecraft ephemeris and attitude models are constructed from the input preprocessed ancillary data by extracting the ancillary data that span the current image. Both ECI and ECEF versions of the ephemeris data are retained in the model structure to avoid the need to repeatedly invoke the ECI/ECEF coordinate system conversion. The ALIAS heritage roll-pitch-yaw representation of the attitude model is retained in the model structure though a quaternion representation may be used in a future algorithm revision (see note 4).

Prepare TIRS LOS Model Sub-Algorithm
This function gathers the information from the preprocessed ancillary data and the Level 0R/1R TIRS image and ancillary data needed to process model data and run the TIRS LOS model. Though it has the same overall purpose and function as the heritage OLI oli_run_model unit, new logic is required to handle the TIRS SSM telemetry information. The spacecraft (preprocessed ancillary data) sections are the same as the OLI model.

The main steps are:

7. Load the image time codes and convert to seconds since spacecraft epoch.
8. Determine the image time window.
9. Validate/smooth the image time codes.
10. Extract the integration time from the Level 0R/1R image line header data.
11. Extract and preprocess the SSM telemetry from the Level 0R ancillary data.
12. Extract the associated ephemeris and attitude data from the preprocessed ancillary data stream.
13. Preprocess the input attitude data into a low-frequency stream, used for basic geometric modeling, and a high-frequency stream, used as a fine correction in the image resampler. This preprocessing was added to improve the ability of the geometric correction system to compensate for jitter disturbance frequencies above 10 Hz.

The input preprocessed ancillary data are stored in an HDF file. The attitude and ephemeris ancillary data streams each have an epoch time identifying the UTC date/time reference. Within these data streams, each attitude or ephemeris observation in the HDF file has a corresponding time offset relative to the epoch. This incoming ancillary data stream spans the entire imaging interval containing the image data represented in the Level 0R/1R input data. In creating the model we identify and extract the ancillary data sequence required to process the current image data.

The input Level 0R/1R image data are also packaged in HDF files that include the image samples for each band and SCA and the time codes assigned to each image line by the TIRS instrument. As shown in figure 2 above and figure 4 below, these spacecraft time codes are provided by the TIRS in

CCSDS T-Field format which includes days since epoch (16-bit integer), milliseconds of day (32-bit integer) and microseconds of millisecond (16-bit integer) fields:

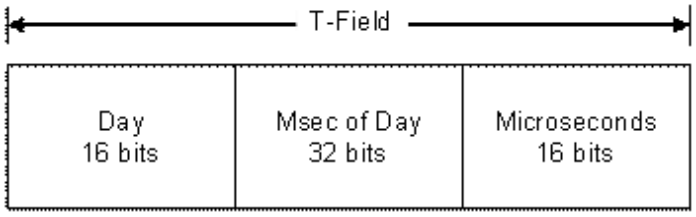


Figure 4: TIRS Time Code Format

The baseline algorithm assumes that Level 0 processing will preserve these time codes in their original form, as days, milliseconds, and microseconds since the spacecraft epoch. Since they are derived from the spacecraft clock, the image time codes will be based on the same epoch used by the ancillary data (e.g., TAI seconds from J2000). Like all fields in the TIRS line header, the time codes are packaged in 12 bit fields with only the low order 8 bits containing valid data (see figure 2). Any initial line header preprocessing steps necessary to extract the 8 valid data bits from each line header data word are assumed to have been performed as part of Level 0Rp generation.

Process Image Time Codes

The image time codes are loaded from the input HDF Level 0R/1R data set. Deselected/replaced detector alignment fill will be inserted into the Level 0R/1R imagery as described above, if necessary, so the image lines each contain samples collected at times that may be offset from the time specified by the corresponding time code. The relationship between these time codes, the TIRS integration time, and the pixel center times has already been described above. The assumption here is that the L0Rp data will contain one time code per image line, excluding any fill lines, or a nominal 2071 time codes per scene. The image files themselves may be up to 30 lines longer to accommodate the redundant row deselect/replacement detector-alignment fill pixels. Simulated scenes may also be longer to provide the additional scene-to-scene overlap needed to support interval stitching.

9. Convert the time code to seconds from spacecraft epoch:
   Line_time = TC_Day*86400 + TC_MSec/1000 + TC_Micro/1e6
   Note that an IEEE 754 double precision (64-bit) number with a 52-bit fraction should provide sufficient precision to represent time differences from 01JAN2000 to 01JAN2050 with microsecond accuracy (1.6e15 microseconds < 2^51).
10. Validate and correct the image time codes as follows:
    a. Loop through the time codes from 1 to N-1, where N is the number of image data frames/time codes, and test the difference between the current and previous time codes against the nominal frame time from the CPF using the #define tolerance DTIME_TOL. The first of two consecutive time codes that are within the tolerance is the first valid time code. The DTIME_TOL value is a constant in the prototype code but it would be better for it to be a CPF parameter.
    b. Initialize the TIRS clock model by setting the least squares variables to zero: $A_{00} = A_{01} = A_{11} = L_0 = L_1 = 0$
        i. Since the normal equation matrix, A, is symmetric, $A_{10} = A_{01}$ so it is not computed separately.
        ii. Add the first valid time code observation by adding 1 to A_{00}. This is all that is required since, by definition, the index difference and time difference (see below) are zero at the first valid point.
    c. For each subsequent time code:

i. Compare the time code difference to a larger outlier tolerance (OUTLIER_TOL) chosen to bound the possible drift in the TIRS clock relative to the spacecraft clock (currently set to 50 microsec). The OUTLIER_TOL value is a constant in the prototype code but it would be better for it to be a CPF parameter.
ii. If the time code difference is within the outlier range, add the current time to a least squares linear TIRS clock model:
    1. Δnum = current index – first valid index
    2. Δtime = current time – first valid time
    3. Accumulate:
        a. Valid point count: A_{00} += 1
        b. Index difference: A_{01} += Δnum
        c. Squared index diff: A_{11} += Δnum*Δnum
        d. Time difference: L_0 += Δtime
        e. Time diff*index diff: L_1 += Δnum*Δtime

d. Once all time codes have been analyzed, solve for the linear TIRS clock model parameters:
    i. determinant = $A_{00}*A_{11} – A_{01}*A_{01}$
    ii. If abs(determinant) <= 0.0 return an error
    iii. Offset = first valid time + $(A_{11}*L_0 – A_{01}*L_1)$ / determinant
    iv. Rate = $(A_{00}*L_1 – A_{01}*L_0)$ / determinant
e. Use the correction model to replace bad time codes:
    i. For each time code:
        1. Calculate the corresponding model time as:
           Mtime = Offset + (current index – first valid index) * Rate
        2. Calculate the actual time – model time difference.
           Diff = abs(time code – Mtime)
        3. Test the difference against DTIME_TOL
        4. If the difference exceeds DTIME_TOL, replace the current time code with the model value, Mtime
f. If no valid time codes were found, return an error.
g. Calculate the average observed frame time, delta_time, by subtracting the first valid/corrected time code from the last valid/corrected time code and dividing by the number of time codes minus one.
h. Store delta_time in the model.

11. Compute the image start time: image_start = line_time[0]
12. Ensure that the ancillary data ephemeris covers the image:
    a. Convert the ephemeris epoch to TAI seconds from spacecraft epoch:
        i. Load the leap second table from the CPF.
        ii. Search the leap second table for the last entry that is not later than the ephemeris epoch.
        iii. Add the total leap second count for that entry to the UTC date/time to yield TAI date/time.
        iv. Subtract the spacecraft TAI epoch to compute ephem_start in TAI seconds since the spacecraft epoch.
    b. Check the beginning of the ephemeris interval against the image start time (which is also TAI seconds since the spacecraft epoch):
       If (image_start – ephem_start) < 4 seconds
       Then report error “Ephemeris data does not cover the image” and exit

c. Check the end of the ephemeris interval against the image stop time:
ephem_stop = ephem_start + ephem_time[M-1] where M is the number of ephemeris data entries.
image_stop = image_start + delta_time*(N-1)
If (ephem_stop – image_stop) < 4 seconds
Then report error “Ephemeris data does not cover the image” and exit
d. Note that the 4 second overlap threshold would be a good thing to put in a #define statement as suggested below.

13. Repeat step 4 using the ancillary attitude data in place of the ephemeris data.
14. Compute the image start UTC epoch by converting image_start to UTC as described under “Convert Spacecraft Time Code to UTC” in the Ancillary Data Preprocessing ADD. This epoch will be stored as: Year, Day of Year, Seconds of Day.
15. Make sure the epoch is consistent with the ancillary data:
    a. If image_year > ephem_year or image_day > ephem_day
    Then image_year = ephem_year
    image_day = ephem_day
    image_seconds = image_seconds + 86400
    This ensures that all computations for a given imaging interval are based on the same day and, hence, on the same UT1UTC, pole wander, and leap second corrections.
16. Subtract the image start time from the line time codes so that the times are seconds from image start.
17. Store the image start UTC epoch (image_year, image_day, image_seconds) and the image offset times in the model structure.
18. Report/trend the results of the time code processing including:
    a. WRS Path/Row (input parameters)
    b. Image UTC epoch (year, day, seconds of day)
    c. L0R ID (input parameter)
    d. Work order ID (input parameter)
    e. Computed frame time (delta_time)
    f. Number of replaced time codes (bad_image_time_count)

Extract the integration time field from the Level 0R/1R image line header data. There will be one value for each image line. Convert the integration time from the first valid TIRS line header record to units of seconds and store in the model structure.

Extract and Process SSM Data

To populate the SSM model portion of the TIRS geometric model it is necessary to construct two elements: the SSM alignment matrix **M’**($\Delta r, \Delta p+2\Delta\gamma, \Delta y$) which is a function of the SSM alignment parameters from the CPF; and a sequence of SSM pointing angles and associated times. The SSM pointing angles are computed using the SSM telemetry data contained in the Level 0R ancillary data. The TIRS ancillary data group is provided every second. The relevant contents of the TIRS ancillary data group are shown in Table 2. Each 1 Hz group contains a time code and twenty-one 24-bit resolution SSM encoder samples.

Field	Size	Contents
Day	16 bits	Days since spacecraft epoch
Milliseconds	32 bits	Milliseconds of day
Microseconds	16 bits	Microseconds of millisecond
SSM Position 1-	24 bits	24-bit resolution SSM encoder readout

21		

Table 2: TIRS Scene Select Mirror Ancillary Data

A twenty-first sample is included, even though the encoder is sampled at 20 Hz, to ensure that encoder samples do not accumulate in the ancillary data output buffer. This means that the twenty-first sample will likely contain zeros in both the high order and low order words. Due to variations in the encoder data generation and ancillary buffer output timing, the high- and low-order encoder data words can become misaligned, and one or both of the data words in sample #21 will be non-zero. This condition must be detected and corrected by the SSM data processing logic.

Retrieve the SSM alignment angle and mirror angle deviation parameters from the CPF and construct the SSM alignment matrix, **M'**, using the equations above. Store the resulting alignment matrix in the SSM model. Determine which mirror control electronics (MCE) side (A or B) is active by performing a majority vote on the MCE bits (bits 0 and 1) of the elec_enabled_flags status word. If the number of status words with bit 1 set exceeds the number with bit 0 set, make the SSM reference angle equal to the CPF side B mirror nadir angle. Otherwise, use the side A value.

Quality check the entire TIRS SSM telemetry set as follows:

1. Validate the SSM telemetry time codes:
    a. Find the first valid time code as the first time code for which the time difference between it and the following time code is the nominal 1.0 second sampling interval, to within a pre-defined tolerance.
    b. Use the valid time code to correct all previous time codes using the nominal sampling interval.
    c. Use the valid time code to check all subsequent time codes using the nominal sampling interval. Any time codes failing the sampling interval tolerance are corrected using the previous valid sample time and the nominal sampling interval.
2. Due to the asynchronous SSM telemetry generation and ancillary data assembly processes, the SSM encoder samples will sometimes be improperly aligned with the 1 Hz ancillary data frames. The encoder telemetry generation logic can run either faster or slower than nominal, leading to variations in the number of samples accumulated in the output buffer between 1 second buffer read operations. Due to these variations in the encoder sample timing, any given ancillary data frame may contain from 19 to 21 encoder samples. The TIRS ancillary data assembly logic was modified to read (and output) 21 samples for each frame to ensure that any extra encoder samples do not accumulate in the encoder telemetry buffer. Furthermore, the upper 16 bits and lower 16 bits of each encoder value are buffered separately, so a given frame may have partial encoder samples (only upper 16 or only lower 16). The following logic is designed to align the upper and lower data words of the SSM telemetry encoder samples in each ancillary data record:
    a. If encoder sample 20 is zero and encoder sample 21 is zero
        i. If this is the last ancillary record, set sample 20 equal to sample 19
        ii. Otherwise set sample 20 equal to sample 1 from the next record.
    b. If the high 16-bits of sample 20 are zero:
        i. If the high 16-bits of sample 21 are non-zero, move the bits from sample 21 to sample 20.
        ii. Otherwise, move the high 16-bits from the first sample in the next record to sample 20, and move all the high order words in the next record up one sample (setting the 21st sample to zero). If the current record is the last record, copy sample 19 into sample 20.

   c. If the low 16-bits of sample 20 are zero:
      i. If the low 16-bits of sample 21 are non-zero, move the bits from sample 21 to sample 20.
      ii. Otherwise, move the low 16-bits from the first sample in the next record to sample 20, and move all the low order words in the next record up one sample. If the current record is the last record, copy sample 19 into sample 20.
   d. If sample 21 is equal to zero, go to the next record.
   e. If only the high 16-bits of sample 21 are zero, and this is not the last record, move all the high order words in the next record up one sample.
   f. If only the low 16-bits of sample 21 are zero, and this is not the last record, move all the low order words in the next record up one sample.
3. Extract the first 20 samples in each record for subsequent processing. Note that this method discards extra samples and fills missing samples by duplicating either the next or the previous sample.
4. Validate the 24-bit SSM telemetry encoder readings:
   a. Find the first valid encoder reading as the first reading for which the difference between it and the SSM reference angle (nominal nadir pointing angle for the current MCE side, determined above) is less than the quality tolerance specified in the CPF.
   b. Use the valid angle to replace all previous 24-bit encoder readings.
   c. Use the valid angle to check and, if necessary, replace all subsequent 24-bit encoder readings, by comparing each value to the previous, valid/corrected, value.
5. Although an anomaly with the SSM encoder that occasionally rendered the upper 14 bits of the read out invalid, was observed during subsystem level test, this behavior has not been seen in the integrated TIRS instrument. Due to the low probability of this being a problem, no special logic has been added to handle this case. Instead, standard outlier detection and correction logic will be relied upon to correct this problem if it occurs.

The quality-checked TIRS ancillary data groups are examined to find the range of samples that correspond to the TIRS image.

1. Scan through the TIRS ancillary data, and convert each time code to seconds from spacecraft epoch.
2. Find the last ancillary data record with a time code that is earlier than the TIRS image start time. This is the first TIRS ancillary data packet to extract.
3. Find the first ancillary data record with a time code that is later than the TIRS image stop time. This is the last TIRS ancillary data packet to extract.
4. Extract the SSM (mirror control electronics or MCE) telemetry fields and time codes for the ancillary data records covering the TIRS image.

For each extracted SSM telemetry group:

1. Convert the TIRS ancillary data time code to seconds from spacecraft epoch and find the difference between the ancillary data time and the TIRS image start time.
2. Add the SSM encoder time offset (from the CPF) to the sample time so that it represents the time of the first SSM encoder sample.
3. Compute the sample times for the remaining samples by adding increments of 0.05 seconds to the previous sample time.
4. Load the SSM encoder samples into signed 32-bit integer variables for subsequent manipulation.
5. Scale the encoder counts to radians and subtract the SSM reference angle (set as described above):
   a. Angle = 2*pi*Sample_Value/0x01000000 – θ_0

   b. If Angle > pi Then Angle -= 2*pi
   c. If Angle < -pi Then Angle += 2*pi

Note: These angles should all be close to zero for Earth viewing.

6. Add the 20 computed times (from image start) and the 20 SSM angles to the SSM model.

Smooth the SSM angles as follows:

1. For each SSM angle:
   a. Find the previous two and next two original data points, or the closest four points if two cannot be found before and after.
   b. Compute the average of the current point and the four closest points.
   c. The smoothed value is the mean of the current point value and the 4 closest points.
2. Store the smoothed sequence of SSM angles in the SSM model.

The baseline TIRS geometric algorithms assume no significant temperature dependence in either the SSM or in TIRS alignment. We do assume that the temperature telemetry present in the TIRS ancillary data will be recorded in the trending database (for radiometric purposes) so that pointing temperature sensitivities could be studied on-orbit (see note #8).

Extract Ancillary Ephemeris and Attitude Data

The subset of ancillary ephemeris and attitude data needed to span the image data are extracted from the input preprocessed ancillary data stream and stored in the model structure. Extra ancillary data, nominally 4 seconds, is required before and after the image start/stop times to ensure model continuity from scene to scene within an imaging interval. This ancillary data overlap time parameter could be stored in a #define statement as it would not be expected to change once established.

The ephemeris data extraction/subsetting procedure is as follows:

7. Compute the time offset from the ephemeris epoch time to the desired ephemeris start time for this image. Note that since the image epoch has been adjusted to fall in the same day as the ephemeris epoch this can be done using the seconds of day fields only.
   ephem_start = image_seconds – ancillary_overlap – ephem_seconds
   Noting that image_seconds and ephem_seconds are the seconds of day fields from the image and ephemeris epoch times, respectively.
8. Loop through the ephemeris sample times to find the last entry that does not exceed ephem_start. This is the ephemeris start index (eph_start_index).
9. Compute the time offset from the ephemeris epoch time to the desired ephemeris stop time for this image.
   ephem_stop = image_seconds + line_time[N-1] + ancillary_overlap – ephem_seconds
   N is the number of image lines, and N-1 is the index of the last image line time.
10. Loop through the ephemeris sample times to find the first entry that exceeds ephem_stop. This is the ephemeris stop index (eph_stop_index).
11. Compute a new ephemeris UTC epoch for this image:
    imgeph_year = ephem_year
    imgeph_day = ephem_day
    imgeph_seconds = ephem_seconds + ephem_samp_time[eph_start_index]
12. Load the ECI and ECEF ephemeris samples from eph_start_index to eph_stop_index (inclusive) into the model structure, adjusting the sample times so that they are offset from the UTC epoch computed in step 5.

The attitude data extraction/subsetting procedure is as follows:

8. Compute the time offset from the attitude epoch time to the desired attitude start time for this image. Note that since the image epoch has been adjusted to fall in the same day as the ancillary data (ephemeris and attitude) epochs this can be done using the seconds of day fields only.
   att_start = image_seconds – ancillary_overlap – att_seconds
   Noting that image_seconds and att_seconds are the seconds of day fields from the image and attitude epoch times, respectively.
9. Loop through the attitude sample times to find the last entry that does not exceed att_start. This is the attitude start index (att_start_index).
10. Compute the time offset from the attitude epoch time to the desired attitude stop time for this image.
    att_stop = image_seconds + line_time[N-1] + ancillary_overlap – att_seconds
11. Loop through the attitude sample times to find the first entry that exceeds att_stop. This is the attitude stop index (att_stop_index).
12. Compute a new attitude UTC epoch for this image:
    imgatt_year = att_year
    imgatt_day = att_day
    imgatt_seconds = att_seconds + att_samp_time[att_start_index]
13. For Earth-view acquisitions, load the roll-pitch-yaw samples from att_start_index to att_stop_index (inclusive) into the model structure, adjusting the sample times so that they are offset from the UTC epoch computed in step 5.
14. For lunar/stellar acquisitions, convert the ECI quaternion samples from att_start_index to att_stop_index (inclusive) to ECI roll-pitch-yaw values, as described below, and store the computed roll-pitch-yaw values in the model structure, adjusting the sample times so that they are offset from the UTC epoch computed in step 5.

Converting ECI Quaternions to Roll-Pitch-Yaw

For lunar and stellar acquisitions, the ECI attitude representation is stored in the model structure. In the baseline model, this is done by converting the ECI quaternions to roll-pitch-yaw values relative to the ECI axes. This is one of the motivations for considering a transition to using a quaternion attitude representation in the model in the future.

The ECI quaternions are converted to roll-pitch-yaw values as follows:

4. Compute the rotation matrix corresponding to the ECI quaternion values:

$$\mathbf{M}_{ACS2ECI} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix}$$

5. Compute the corresponding ACS to ECI roll-pitch-yaw values:

$$\text{roll}' = -\tan^{-1}\left(\frac{M_{2,1}}{M_{2,2}}\right)$$

$$\text{pitch}' = \sin^{-1}\left(M_{2,0}\right)$$

$$\text{yaw}' = -\tan^{-1}\left(\frac{M_{1,0}}{M_{0,0}}\right)$$

Note that in implementing these calculations it is important to use the ATAN2 rather than the ATAN arctangent implementation in order to retain the correct quadrants for the Euler angles. This is not a concern in Earth-view imagery where the angles are always small, but becomes an issue for these lunar/stellar ACS to ECI angles.

6. Store the ECI roll-pitch-yaw values in the model attitude data table.

At the completion of this sub-algorithm the model structure contains the image frame time stamps, the multispectral and panchromatic sample and integration times, the ancillary ephemeris data, in both ECI and ECEF representations, covering the image, and the ancillary attitude data covering the image.

Jitter Correction Data Preprocessing

Jitter correction preprocessing operates on the roll-pitch-yaw attitude data stream extracted from the spacecraft ancillary data to separate the low frequency spacecraft pointing effects from the higher frequency jitter disturbances. The low frequency pointing model is used for line-of-sight projection and other geolocation processing while the high frequency jitter effects are applied as per-line corrections during image resampling. To implement this frequency separation in the line-of-sight model the original attitude sequence is passed through a low pass filter with a cutoff frequency defined as a parameter in the CPF. This cutoff frequency will nominally be in the 1 Hz to 10 Hz range. The value ultimately selected for this cutoff frequency will depend upon the actual disturbance profile observed in the spacecraft attitude data. The high frequency data stream should be limited in magnitude to sub-pixel (ideally sub-half-pixel) effects, but the lower the cutoff frequency can be, the sparser (and smaller) the TIRS resampling grid can be made in the line (time) dimension.

The low pass filtered version of the attitude sequence is differenced with the original data to construct the complementary high pass data sequence. The high pass sequence is then interpolated at the TIRS image line times to provide a table containing high frequency roll-pitch-yaw corrections for each image line. This jitter table is stored in the TIRS line-of-sight model. The original attitude sequence in the line-of-sight model is replaced with the low pass filtered sequence to avoid double counting the high frequency effects. This process is depicted in figure 5.

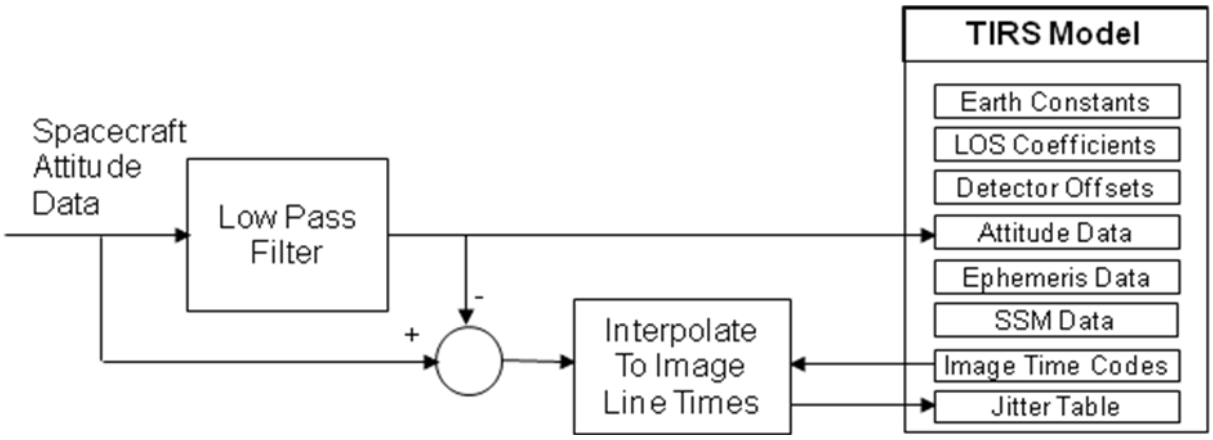


Figure 5: Jitter Correction Table Generation Data Flow

The jitter table construction processing sequence is as follows:

7. Extract a copy of the original attitude data sequence from the TIRS line-of-sight model.
8. Retrieve the low pass filter cutoff frequency from the CPF.
9. Design a low pass filter with the desired cutoff frequency and apply it to the attitude data.
    a. Use the cutoff frequency and attitude data sampling time to compute the size of the desired filter as follows:
        i. Compute the normalized cutoff frequency (the ratio of the cutoff frequency to the attitude data sampling frequency):
            n_cutoff = cutoff_frequency / attitude_sample_frequency
        Note that this is the same as:
            n_cutoff = cutoff_frequency * attitude_sample_time
        ii. Compute the number of samples per cycle at the cutoff frequency:
            Nsamp = 1 / n_cutoff
        iii. Multiply the number of samples per cycle by 3 and add 1 to yield the desired filter size:
            FSize = 3*Nsamp + 1
        iv. If this results in an even filter size, add one:
            If (FSize modulo 2 == 0) FSize = FSize + 1
    b. Use the Remez exchange algorithm to design the filter and generate the filter weights. The standard Parks-McClellan finite impulse response (FIR) digital filter design method uses the Remez exchange algorithm (ref. Theory and Application of Digital Signal Processing, Rabiner and Gold, Prentice-Hall, 1975). A C implementation of this algorithm called remez.c, authored by Jake Janovetz at the University of Illinois, is available under the GNU Public License. This implementation specifies the desired (low pass, in this case) filter response using the following parameters:
        i. Filter size (number of taps) – FSize computed in item a. above.
        ii. Number of frequency bands to use – 2, one pass band (low frequency) and one stop band (high frequency).
        iii. Band frequency bounds – 0 to the normalized cutoff frequency (n_cutoff) for the pass band and 1.5*n_cutoff to 0.5 (normalized Nyquist frequency) for the stop band.
        iv. Desired band gains – 1 for pass band (low) and 0 for stop band (high).
        v. Band weights (how tightly to constrain the actual filter response to the design filter response in each band) – 1 for pass band and 10 for stop band.

vi. Filter type – BANDPASS (the low pass filter is a special case of the more general BANDPASS filter type supported by the remez algorithm.

c. Make sure the synthesized filter is normalized (weights sum to 1) by adding the filter tap values and dividing each tap by the total.

sum = Σ h[i] where h[i] are the FSize filter taps.
h'[i] = h[i] / sum for i = 1 to FSize.

d. Convolve the filter with the roll-pitch-yaw attitude data one axis at a time:

```
half_size = FSize / 2
for index = 0 to num_rpy – 1
    low_roll[index] = low_pitch[index] = low_yaw[index] = 0
    for ii = -half_size to half_size
        if ( index + ii < 0 ) j = -index – ii
        else if ( index + ii < num_rpy ) j = index + ii
        else j = 2*num_rpy – index - ii – 1
    low_roll[index] += roll[j]*h[ii+half_size]
    low_pitch[index] += pitch[j]*h[ii+half_size]
    low_yaw[index] += yaw[j]*h[ii+half_size]
```

10. Subtract the low pass filtered sequences from the original sequences to extract the high frequency portion of the data, and transfer any residual bias (non-zero mean value) from the imaging portion of the high frequency sequence to the low frequency sequence:

```
roll_bias = pitch_bias = yaw_bias = 0
att_pts = 0
for index = 0 to nrpy–1
    high_roll[index] = roll[index] – low_roll[index]
    high_pitch[index] = pitch[index] – low_pitch[index]
    high_yaw[index] = yaw[index] – low_yaw[index]
    if ( image_start_time < attitude_time[index] < image_stop_time )
        roll_bias += high_roll[index]
        pitch_bias += high_pitch[index]
        yaw_bias += high_yaw[index]
        att_pts += 1
roll_bias = roll_bias / att_pts
pitch_bias = pitch_bias / att_pts
yaw_bias = yaw_bias / att_pts
for index = 0 to nrpy-1
    high_roll[index] -= roll_bias
    low_roll[index] += roll_bias
    high_pitch[index] -= pitch_bias
    low_pitch[index] += pitch_bias
    high_yaw[index] -= yaw_bias
    low_yaw[index] += yaw_bias
```

11. Interpolate the high frequency sequence values at the TIRS line sampling times to create the model jitter table:

```
For each TIRS image line = 0 to number of lines:
    Compute the line sampling time as:
        index = line
        line_time = line_time_stamp[index]
                  + integration_time/2
    Convert to time from attitude epoch:
```

line_time += image_epoch – attitude _epoch

Interpolate high frequency roll-pitch-yaw values at this time using four point Lagrange interpolation:

Compute starting index for interpolation:
index = floor(line_time / attitude_sample_time) – 1
Compute the fractional sample offset to the line time:
w = line_time / attitude_sample_time – index – 1
Compute the Lagrange weights:
w1 = -w * (w – 1) * (w – 2) / 6
w2 = (w + 1) * (w – 1) * (w – 2) / 2
w3 = -w * (w + 1) * (w – 2) / 2
w4 = (w + 1) * w * (w – 1) / 6
Interpolate:
roll = high_roll[index]*w1 + high_roll[index+1]*w2
+ high_roll[index+2]*w3 + high_roll[index+3]*w4
pitch = high_pitch[index]*w1 + high_pitch[index+1]*w2
+ high_pitch[index+2]*w3 + high_pitch[index+3]*w4
yaw = high_yaw[index]*w1 + high_yaw[index+1]*w2
+ high_yaw[index+2]*w3 + high_yaw[index+3]*w4

12. Replace the original model attitude data sequence with the low pass filtered attitude data sequence.

Note that if TIRS and OLI processing is combined, the attitude filtering and high-pass/low-pass separation logic should be common, but the two sensors would still require their own jitter tables since these tables are based on the image line times, which are different for TIRS and OLI.

Process LOS Model Sub-Algorithm

This function loads the LOS Legendre polynomial coefficients and other model components from the CPF, and performs additional processing on the attitude and ephemeris information in the LOS model structure. It invokes the following sub-algorithms.

Read CPF Model Parameters Sub-Algorithm

This function loads model components from the CPF. In the heritage ALIAS implementation some of these model components either did not exist (e.g., instrument offset from spacecraft center of mass) or were used for image resampling but not LOS model computations (e.g., detector offset table) and so, were not included in the model. These are included in the TIRS model to make it self-contained for purposes of line-of-sight computations.

CPF parameters loaded into the geometric model include:

7. Earth orientation parameters – the UT1UTC and pole wander (x,y) parameters for the current day are stored in the model to avoid the necessity of repeatedly looking them up in the CPF.
8. TIRS offset from spacecraft center of mass – a 3-vector that captures the small offset, in spacecraft body coordinates, between the TIRS instrument, where images are captured, and the spacecraft center of mass, the position of which is reported in the ancillary ephemeris data, making it possible to translate the ephemeris data to the TIRS. Technically, this would be the vector from the spacecraft center of mass to the center of the TIRS entrance pupil. Note that this formulation assumes that the spacecraft on-board GPS data processing includes the GPS to spacecraft center of mass (CM) offset and that the spacecraft is, in fact, reporting CM

positions not GPS antenna positions. If the ephemeris represents the GPS antenna location then we would need to know the spacecraft CM to GPS antenna offset as well.

9. TIRS to attitude control system (ACS) alignment matrix – a 3-by-3 matrix that captures the relative orientation of the TIRS coordinate system to the ACS coordinate system, making it possible to rotate the TIRS instrument-space line-of-sight vectors into the ACS reference system. In the heritage ALIAS system this was actually represented in the CPF by an ACS to instrument rotation matrix which was inverted for each LOS model invocation. Whichever convention is used in the CPF, the LOS model should store the TIRS-to-ACS rotation matrix.
10. TIRS line-of-sight Legendre polynomials – a set of 8 coefficients (4 along-track and 4 across-track) for each band on each SCA. Each set of 4 forms a 3rd order Legendre polynomial that is used to evaluate a nominal LOS angle (along- or across-track) for the detectors in that band on that SCA. This differs from the heritage implementation which used a 2nd order model (see the Read LOS Vectors Sub-Algorithm description below).
11. TIRS detector delay table – a table consisting of two values (along- and across-track) per detector reflecting the offset of each actual detector from its nominal location (as modeled by the 3rd order Legendre polynomials – see below). In the heritage ALIAS implementation these were small sub-pixel offsets that were applied in the image resampling procedure. With the TIRS, this table will also contain any offsets due to detector deselect/replacement (i.e., the operational use of a detector from one of the redundant rows). This table is needed in those LOS projection algorithms that utilize either actual (whole pixel offsets) or exact (full sub-pixel offsets) detector locations.

Read LOS Vectors Sub-Algorithm

This function retrieves the line of sight vectors from the CPF. The line of sight vectors are stored as sets of 3rd order Legendre polynomial coefficients. There is a unique set of 8 coefficients for each band of each SCA, 4 for the along-track polynomial and 4 for the across-track polynomial. These values are read from the CPF and stored in the LOS model. The polynomials are used to compute along- and across-track viewing angles for each nominal detector.

Initialize the Precision Model Sub-Algorithm

This function initializes the precision LOS correction model parameters. If the optional precision model input parameters are provided, those values are used. In the normal case, those parameters are absent and the correction model is initialized as follows:

Set the precision correction reference time to the center of the scene:
t_ref = line_time[N/2] where: N is the number of time codes in the image
Set the ephemeris correction model order to zero: eph_order = 0
Set both ephemeris X correction parameters to zero:
x_corr[0] = 0.0, x_corr[1] = 0.0
Set both ephemeris Y correction parameters to zero:
y_corr[0] = 0.0, y_corr[1] = 0.0
Set both ephemeris Z correction parameters to zero:
z_corr[0] = 0.0, z_corr[1] = 0.0
Set the attitude correction model order to zero: att_order = 0
Set all three attitude roll correction parameters to zero:
roll_corr[0] = 0.0, roll_corr[1] = 0.0, roll_corr[2] = 0.0
Set all three attitude pitch correction parameters to zero:
pitch_corr[0] = 0.0, pitch_corr[1] = 0.0, pitch_corr[2] = 0.0
Set all three attitude yaw correction parameters to zero:
yaw_corr[0] = 0.0, yaw_corr[1] = 0.0, yaw_corr[2] = 0.0

Note that these parameters are used to compute the corrected ephemeris and attitude data sequences which are also stored in the model. The parameters themselves are included in the model primarily to document the magnitude of the corrections applied and to facilitate more advanced uses of the model creation logic. For example, it is sometimes useful to be able to force a particular model bias (e.g., a roll angle) into a model that is to be used for data simulation (see note 6). So, though not strictly necessary for operational data processing, these parameters aid in anomaly resolution, data simulation, and algorithm development. In normal operations, these initial correction parameters are all zero and the "corrected" attitude and ephemeris data sequences are identical to the "original" attitude and ephemeris data prior to the execution of the LOS model correction algorithm. Subsequent algorithms (e.g., LOS projection) operate on the corrected data.

Correct Attitude Sub-Algorithm

This function applies the ACS/body space attitude corrections computed by the LOS/precision correction procedure to the attitude data sequence. It outputs a parallel table of roll-pitch-yaw values with the precision corrections applied. In the model creation context the precision corrections are zero so the two sets of attitude data are identical. Though applying the precision corrections to construct the corrected attitude sequence could be said to be overkill for model creation (since the corrections are nominally zero at this point) this capability is required for LOS model correction and is used here to support the use of the model creation algorithm for data simulation and anomaly resolution as it makes it possible to force initial biases into the model. This sub-algorithm will also be used by the LOS/precision correction algorithm to create the precision model. Note that the formulation is somewhat different for Earth-view scenes (Acquisition Type = Earth) than it is for lunar and stellar observations.

Earth Scenes

For Earth-view scenes the sequence of transformations required to convert a line-of-sight in the TIRS instrument coordinate system, generated using the Legendre polynomials, is:

$$\underline{\mathbf{x}}_{ECEF} = \mathbf{M}_{ORB2ECEF}\,\mathbf{M}_{ACS2ORB}\,\mathbf{M}_{Precision}\,\mathbf{M}_{TIRS2ACS}\,\mathbf{M}_{SSM}(\theta)\,\mathbf{M}_{Tele2SSM}\,\underline{\mathbf{x}}_{TIRS}$$

where:

$\underline{\mathbf{x}}_{TIRS}$ is the Legendre-derived instrument LOS vector
$\mathbf{M}_{Tele2SSM}$ is the TIRS telescope alignment matrix described above
$\mathbf{M}_{SSM}(\theta)$ is the SSM reflection matrix, described above, which is a function of SSM angle
$\mathbf{M}_{TIRS2ACS}$ is the TIRS to ACS alignment matrix from the CPF
$\mathbf{M}_{Precision}$ is the correction to the attitude data computed by the LOS/precision correction procedure
$\mathbf{M}_{ACS2ORB}$ is the spacecraft attitude (roll-pitch-yaw)
$\mathbf{M}_{ORB2ECEF}$ is the orbital to ECEF transformation computed using the ECEF ephemeris
$\underline{\mathbf{x}}_{ECEF}$ is the LOS vector in ECEF coordinates

Since TIRS will occasionally be viewing off-nadir and it is more natural to model attitude errors in the ACS/body coordinate system than in the orbital coordinate system, the order of the $\mathbf{M}_{ACS2ORB}$ and $\mathbf{M}_{Precision}$ rotations have been reversed for LDCM as compared to the heritage Landsat/EO-1 implementation. The impact is minimal in the model and LOS projection but becomes more important for the LOS/precision correction algorithm.

This new sub-algorithm pre-computes the $\mathbf{M}_{ACS2ORB}$ $\mathbf{M}_{Precision}$ combination and stores the corresponding corrected roll-pitch-yaw attitude sequence in the model structure. This approach has several advantages:

7. It streamlines the application of the model for LOS projection by removing the step of explicitly applying the precision correction.
8. It allows for the use of a more complex correction model in the future since the application of the model is limited to this unit. Note that the Earth-view attitude correction model consists of the following model parameters:
   Precision reference time: t_ref in seconds from the image epoch (at the center of the image time window)
   Attitude model order: att_order = 1
   Roll bias and rate corrections: roll_corr[] = roll_bias, roll_rate
   Pitch bias and rate corrections: pitch_corr[] = pitch_bias, pitch_rate
   Yaw bias and rate corrections: yaw_corr[] = yaw_bias, yaw_rate
   This model is dealt with in more detail in the line-of-sight correction algorithm description.
9. Retaining both the original and corrected attitude sequences in the model make the model self-contained and will make it unnecessary for the LOS/precision correction algorithm to access the preprocessed ancillary data.

The disadvantage is that it doubles the size of the attitude data in the model structure.

The construction of the corrected attitude sequence proceeds as follows:

For each point in the attitude sequence j = 0 to K-1:

10. Compute the rotation matrix corresponding to the j^{th} roll-pitch-yaw values:

$\mathbf{M}_{ACS2ORB}$ =

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

11. Compute the precision correction at the time (t_att = att_seconds + att_time[j]) corresponding to the attitude sample:

a. $$roll\_correction = \sum_{i=0}^{att\_order} \mathbf{roll\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$$

b. $$pitch\_correction = \sum_{i=0}^{att\_order} \mathbf{pitch\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$$

c. $$yaw\_correction = \sum_{i=0}^{att\_order} \mathbf{yaw\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$$

Note that only the seconds of day fields are needed for the attitude and image epochs as they are constrained to be based on the same year and day.

12. Compute the rotation matrix corresponding to roll_correction (r), pitch_correction (p), and yaw_correction (y) ($\mathbf{M}_{Precision}$) using the same equations presented in step 1 above.
13. Compute the composite rotation matrix: $\mathbf{M} = \mathbf{M}_{ACS2ORB}\ \mathbf{M}_{Precision}$
14. Compute the composite roll-pitch-yaw values:

$$\text{roll}' = -\tan^{-1}\left(\frac{M_{2,1}}{M_{2,2}}\right)$$

$$\text{pitch}' = \sin^{-1}\left(M_{2,0}\right)$$

$$\text{yaw}' = -\tan^{-1}\left(\frac{M_{1,0}}{M_{0,0}}\right)$$

15. Store the composite roll'-pitch'-yaw' values in the j^{th} row of the corrected attitude data table.

Lunar and Stellar Scenes

Though there is no TIRS requirement for lunar or stellar data processing, this capability is retained to maintain compatibility with the OLI geometric model. For celestial (lunar or stellar) observations the sequence of transformations required to convert a line-of-sight in the TIRS instrument coordinate system, generated using the Legendre polynomials, is:

$$\underline{\mathbf{x}}_{ECI} = \mathbf{M}_{ACS2ECI}\ \mathbf{M}_{Precision}\ \mathbf{M}_{TIRS2ACS}\ \mathbf{M}_{SSM}(\theta)\ \mathbf{M}_{Tele2SSM}\ \underline{\mathbf{x}}_{TIRS}$$

where:

$\underline{\mathbf{x}}_{TIRS}$ is the Legendre-derived instrument LOS vector
$\mathbf{M}_{Tele2SSM}$ is the TIRS telescope alignment matrix described above
$\mathbf{M}_{SSM}(\theta)$ is the SSM reflection matrix, described above, which is a function of SSM angle
$\mathbf{M}_{TIRS2ACS}$ is the TIRS to ACS alignment matrix from the CPF
$\mathbf{M}_{Precision}$ is the correction to the attitude data computed by the LOS/precision correction procedure
$\mathbf{M}_{ACS2ECI}$ is the spacecraft attitude in the ECI frame derived from the ECI quaternions in the preprocessed ancillary data
$\underline{\mathbf{x}}_{ECI}$ is the LOS vector in ECI coordinates

The advantage of modeling the precision attitude corrections in ACS rather than orbital coordinates becomes apparent here, since the orbital frame is not used in the lunar case.
This sub-algorithm pre-computes the $\mathbf{M}_{ACS2ECI}\ \mathbf{M}_{Precision}$ combination and stores the corresponding corrected attitude sequence (as roll-pitch-yaw values relative to ECI) in the model structure. Another difference between the Earth-view and lunar/stellar models is in the formulation of the precision model. The lunar attitude correction model adds an acceleration term to the Earth-view correction model parameters:

Precision reference time: t_ref in seconds from the image epoch (nominally near the center of the image time window)
Attitude correction model order: att_order = 2
Roll bias, rate, and acceleration corrections: roll_corr[] = roll_bias, roll_rate, roll_acceleration
Pitch bias, rate, and acceleration corrections: pitch_corr[] = pitch_bias, pitch_rate, pitch_acceleration
Yaw bias, rate, and acceleration corrections: yaw_corr[] = yaw_bias, yaw_rate, yaw_acceleration

Due to the different orders of the Earth-view and lunar correction models, this model is stored as an array in the model structure along with a field defining the model order. The precision model is dealt with in more detail in the line-of-sight correction algorithm description.

The processing steps to construct the corrected attitude sequence is the same for lunar/stellar acquisitions, although the interpretation of the roll-pitch-yaw values is slightly different, and proceeds as follows:

For each point in the attitude sequence j = 0 to K-1:

6. Compute the rotation matrix corresponding to the j[th] ECI roll-pitch-yaw values:

$\mathbf{M}_{ACS2ECI}$ =

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

16. Compute the precision correction at the time (t_att = att_seconds + att_time[j]) corresponding to the attitude sample:

a. $roll\_correction = \sum_{i=0}^{att\_order} \mathbf{roll\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$

b. $pitch\_correction = \sum_{i=0}^{att\_order} \mathbf{pitch\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$

c. $yaw\_correction = \sum_{i=0}^{att\_order} \mathbf{yaw\_corr}[i]*(t\_att - t\_ref - image\_seconds)^i$

Note that only the seconds of day fields are needed for the attitude and image epochs as they are constrained to be based on the same year and day.

7. Compute the rotation matrix corresponding to roll_correction (r), pitch_correction (p), and yaw_correction (y):

$\mathbf{M}_{Precision}$ =

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

8. Compute the composite rotation matrix: $\mathbf{M} = \mathbf{M}_{ACS2ECI}\ \mathbf{M}_{Precision}$
9. Compute the composite ACS to ECI roll-pitch-yaw values:

$$\text{roll}' = -\tan^{-1}\left(\frac{M_{2,1}}{M_{2,2}}\right)$$

$$\text{pitch}' = \sin^{-1}\left(M_{2,0}\right)$$

$$\text{yaw}' = -\tan^{-1}\left(\frac{M_{1,0}}{M_{0,0}}\right)$$

Note that in implementing these calculations it is important to use the ATAN2 rather than the ATAN arctangent implementation in order to retain the correct quadrants for the Euler angles. This is not a concern in Earth-view imagery where the angles are always small, but becomes an issue for these lunar/stellar ACS to ECI angles.

10. Store the composite roll'-pitch'-yaw' values in the j[th] row of the corrected attitude data table.

Correct Ephemeris Sub-Algorithm

The heritage ALIAS function converts the ephemeris information (position and velocity) from the Earth Centered Inertial (ECI J2000) system to the Earth Centered Earth Fixed (ECEF) system and applies the ephemeris corrections computed in the LOS/precision correction procedure to both ephemeris

sets. Since both ECI and ECEF representations of the ephemeris are now provided by the ancillary data preprocessing algorithm, the first portion of the heritage algorithm is no longer necessary (or could be reused in the ancillary data preprocessing algorithm). Though applying the precision corrections to construct the corrected ephemeris sequence could be said to be overkill for model creation (since the corrections are nominally zero at this point) this capability is required for LOS model correction and is used here to support the use of the model creation algorithm for data simulation and anomaly resolution as it makes it possible to force initial biases into the model. This sub-algorithm will also be used by the LOS/precision correction algorithm to create the precision model.

The precision correction parameters are stored in the LOS model in the spacecraft orbital coordinate system as three position (x_bias, y_bias, z_bias) corrections and three velocity (x_rate, y_rate, z_rate) corrections that, like the attitude corrections, are relative to t_ref. These values must be converted to the ECEF and ECI coordinate systems. Once the precision correction is determined in the ECEF/ECI coordinate system, the ECEF/ECI ephemeris values can be updated with the precision parameters.

Loop on LOS model ephemeris points j = 0 to N-1

Compute the precision correction:

Calculate delta time for precision correction:

dtime = ephem_seconds + ephem_time[j] – t_ref – image_seconds

Calculate the change in X, Y, Z due to precision correction. Corrections are in terms of spacecraft orbital coordinates.

dx orb = model precision x_corr[0] + model precision x_corr[1] * dtime
dy orb = model precision y_corr[0] + model precision y_corr[1] * dtime
dz orb = model precision z_corr[0] + model precision z_corr[1] * dtime

where:
model precision x_corr[0] = precision (orbital) update to X position
model precision y_corr[0] = precision (orbital) update to Y position
model precision z_corr[0] = precision (orbital) update to Z position
model precision x_corr[1] = precision (orbital) update to X velocity
model precision y_corr[1] = precision (orbital) update to Y velocity
model precision z_corr[1] = precision (orbital) update to Z velocity

Construct precision position and velocity “delta” vectors.

$$[\text{dorb}] = \begin{bmatrix} \text{dx orb} \\ \text{dy orb} \\ \text{dz orb} \end{bmatrix}$$

$$[\text{dvorb}] = \begin{bmatrix} \text{model precision x\_corr[1]} \\ \text{model precision y\_corr[1]} \\ \text{model precision z\_corr[1]} \end{bmatrix}$$

Calculate the orbit to ECF transformation [ORB2ECEF] using ECEF ephemeris (See the ancillary data preprocessing ADD for this procedure).

Transform precision "delta" vectors to ECEF.

$$[\text{def}] = [\text{ORB2ECEF}][\text{dorb}]$$

$$[\text{dvef}] = [\text{ORB2ECEF}][\text{dvorb}]$$

Adjust ECEF ephemeris by the appropriate "delta" precision vector and store the new ephemeris in the model. These ephemeris points will be used when transforming an input line/sample to an output projection line/sample.

$$\text{model ef postion} = \text{ephemeris ecef postion} + \text{decf}$$

$$\text{model ef velocity} = \text{ephemeris ecef velocity} + \text{dvecf}$$

where:

All parameters are 3x1 vectors
ephemeris ecef values are the interpolated one-second ephemeris values in ECEF coordinates

Calculate the orbit to ECI transformation [ORB2ECI] using ECI ephemeris.

Transform precision "delta" vectors to ECI.

$$[\text{deci}] = [\text{ORB2ECI}][\text{dorb}]$$

$$[\text{dveci}] = [\text{ORB2ECI}][\text{dvorb}]$$

Adjust ECI ephemeris by the appropriate "delta" precision vector and store the new ephemeris in the model. These ephemeris points will be used with lunar/stellar observations.

$$\text{model eci postion} = \text{ephemeris eci postion} + \text{deci}$$

$$\text{model eci velocity} = \text{ephemeris eci velocity} + \text{dveci}$$

where:

All parameters are 3x1 vectors
ephemeris eci values are the interpolated one-second ECI ephemeris

Move Satellite Sub-Algorithm

This function computes the satellite position and velocity at a delta time from the ephemeris reference time using Lagrange interpolation. This is a utility sub-algorithm that accesses the model ephemeris data to provide the TIRS position and velocity at any specified time. Since the model ephemeris arrays are inputs to this sub-algorithm it will work with either the ECI or ECEF ephemeris data.

Table 3 below summarizes the contents of the TIRS LOS model structure. The estimated size of this structure is approximately 1.5 megabytes.

LOS Model Structure Contents
Satellite Number (8)
Format Version Number (for documentation and backward compatibility)
WRS Path
WRS Row (may be fractional)
Acquisition Type (Earth, Lunar, Stellar)
Earth Orientation Parameters
UT1UTC Correction (in seconds)
Pole Wander X Correction (in arc seconds)
Pole Wander Y Correction (in arc seconds)
Image Model
Number of image lines
Image UTC epoch: image_year, image_day, image_seconds
For each line: frame time offset (in seconds) from image epoch
For each line: roll, pitch, yaw high frequency jitter correction (in radians)
Nominal alignment fill table (from CPF) one value per band per SCA (in pixels)
Detector alignment fill table (from L0R/L1R) one value per detector (in pixels)
Sensor Model
TIRS to ACS reference alignment matrix [3x3]
Spacecraft center of mass to TIRS offset in ACS reference frame [3x1] in meters
Integration Time in seconds
Computed Sample Time in seconds
Number of SCAs (3)
Number of Bands (4)
Along-Track IFOV in radians
Across-Track IFOVs (MS and pan) in radians
Number of Detectors per SCA in each Band (4x1 array)
Focal plane model parameters (Legendre coefs) [NSCAxNBANDx2x4] (in radians)
Detector delay table [NSCAxNBANDx2xNDET] (in pixels)
Scene Select Mirror Model
Telescope to SSM alignment matrix [3x3]
Number of SSM encoder angles
Time from image epoch (one per sample, nominally 20 Hz) (in seconds)
SSM angle (one per sample) (in radians)
Ephemeris Model
Scene ephemeris data UTC epoch: imgeph_year, imgeph_day, imgeph_seconds
Number of ephemeris samples
Time from epoch (one per sample, nominally 1 Hz) (in seconds)
Original ECI position estimate (X, Y, Z) (one set per sample) (in meters)
Original ECI velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec)
Original ECEF position estimate (X, Y, Z) (one set per sample) (in meters)
Original ECEF velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec)
Corrected ECI position estimate (X, Y, Z) (one set per sample) (in meters)
Corrected ECI velocity estimate (Vx, Vy, Vz) (one set per sample) (in meters/sec)
Corrected ECEF position estimate (X, Y, Z) (one set per sample) (in meters)
Corrected ECEF velocity estimate (Vx, Vy, Vz) (one set per sample) (in

meters/sec)
Attitude Model
Scene attitude data UTC epoch: imgatt_year, imgatt_day, imgatt_seconds
Number of attitude samples
Time from epoch (one per sample, nominally 50 Hz) (in seconds)
Original Roll, pitch, yaw estimate (one per sample) (in radians)
Corrected Roll, pitch, yaw estimate (one per sample) (in radians)
Precision Correction Model
Precision reference time (t_ref) seconds from image epoch
Ephemeris correction order: eph_order (0 none, 1 for Earth-view and lunar/stellar)
X correction model: x_bias, x_rate (meters, meters/sec)
Y correction model: y_bias, y_rate (meters, meters/sec)
Z correction model: z_bias, z_rate (meters, meters/sec)
Attitude correction order: att_order (0 none, 1 for Earth, 2 for lunar/stellar)
Roll correction model: roll_bias, roll_rate, roll_acc (rad, rad/sec, rad/sec^2)
Pitch correction model: pitch_bias, pitch_rate, pitch_acc (rad, rad/sec, rad/sec^2)
Yaw correction model: yaw_bias, yaw_rate, yaw_acc (rad, rad/sec, rad/sec^2)

Table 3: TIRS LOS Model Structure Contents

Note that in the precision correction model only the correction model array elements up to att_order are valid. For example, for Earth-view scenes att_order = 1 and roll_corr[0] = roll_bias, roll_corr[1] = roll_rate and roll_corr[2] is not used.

7.3.1.8 Maturity

Though much of the OLI model creation algorithm was reusable for TIRS there are several areas where changes were necessary:

1. The TIRS SSM telemetry is extracted from the ancillary data stream, quality checked, and smoothed. The SSM model is a new component that has been added to the TIRS LOS model. Known issues with the performance of the SSM encoder may necessitate the development of additional correction logic to account for errors in the high-order encoder bits (see also note #7 below). This is not included in the current baseline.
2. Analysis of the TIRS optical model has shown that the 2nd order Legendre polynomial model used to generate OLI lines-of-sight will not be adequate for TIRS. This is due primarily to the larger field of view of the TIRS SCAs. Since each SCA covers a larger portion of the instrument's field of view, it is subject to more of the optical distortion variations that occur across the field of view. Initial analysis indicates that a 3rd order Legendre model will capture the nominal TIRS detector lines of sight for each band and each SCA with sufficient fidelity. The TIRS detector line-of-sight model has been updated accordingly.
3. Some features of the OLI instrument (e.g., odd-even detector offset) are not relevant for TIRS but are retained, with appropriate calibration parameters set to zero, to maintain commonality across the models.
4. Temperature measurements at the TIRS mount points and in the scene select mirror mechanism may allow the inclusion of temperature dependent effects in the TIRS to ACS (and TIRS to OLI) alignment and/or the SSM model. This is not included in the baseline model but we do assume that the relevant temperature telemetry is trended (see note #8 below).

7.3.1.9 Notes

Some additional background assumptions and notes include:

1. The static precession, nutation, and sidereal time parameters needed to convert Earth Centered Inertial J2000 to/from WGS84 Earth Fixed are built into the software rather than being provided as input data. The dynamic terms (UT1UTC correction, polar wander) are provided in the CPF. For LDCM, the CPF has been expanded to include a leap second table to allow for converting spacecraft TAI-reference time codes to UTC. This TAI to UTC time conversion and the ECI/ECEF conversion algorithm are discussed in the ancillary data preprocessing algorithm description document.
2. While it seems to be generally agreed that the TIRS Level 0R/Level 1R data will not use fill pixels to nominally align bands and/or SCAs, it may include fill pixels to achieve nominal detector-to-detector alignment in the case where bad detectors are replaced from the redundant detector row. Since this is an existing capability in the heritage OLI logic, the L0R/L1R detector alignment fill table input identified in the input table will be retained as the mechanism for the Level 0R/1R data to identify the number of any fill pixels used. In practice, it may be preferable to keep the TIRS L0R in strict time order (with no fill) even if dead detector replacement is performed.
3. The "thresholds and limits" parameters, stored either in system tables or the database for L7 and ALI, will be included in the CPF for LDCM. This will make date specific changes, e.g., due to a change in the nominal orbit during early- or late-mission operations, easier to manage.
4. The current algorithm baseline is to use the heritage attitude model roll-pitch-yaw representation. This could be updated in a future revision to use a quaternion representation. This is the motivation for including both quaternion and roll-pitch-yaw representations of the attitude data sequence in the output from the ancillary data preprocessing algorithm.
5. This algorithm includes a simple image time code validation/smoother function to fix errors and/or smooth out quantization effects in the downlinked time codes. This may not be necessary or it may need to be more elaborate depending on the reliability of the TIRS time codes.
6. The baseline algorithm prototype implementation allows the precision correction model parameters to be provided as optional input parameters. This would not be used for operational data processing and these parameters would not ordinarily be provided, with their values defaulting to those set in the Initialize the Precision Model sub-algorithm. Having such an R&D capability to force model corrections at model creation time can prove useful in applications such as data simulation and anomaly resolution.
7. The reliability of the SSM encoder telemetry is unknown. As a contingency, the baseline model includes a quality check and smoothing logic to allow for SSM encoder data preprocessing. The quality check is based on a threshold check that ensures sample-to-sample consistency. The threshold is a CPF parameter. A check for consistency with a nominal value may also be required.
8. The baseline model does not include any temperature dependent effects in either the SSM or in the overall TIRS alignment. The temperature telemetry provided in the TIRS ancillary data would provide a means for investigating any such dependencies on orbit. This algorithm assumes that the TIRS temperature telemetry will be collected and trended by the radiometric processing algorithms and would therefore be available, if needed, for future implementation of temperature-based calibration adjustments.
9. The SSM encoder position is provided at a 20 Hz sampling rate even though the SSM telemetry packets are generated at 1 Hz. Each packet contains 20 samples, with each sample representing one 24-bit encoder read out.
10. The TIRS detector deselect mechanism and detector offset geometry differ from the OLI versions but the same correction logic can be applied. In the OLI case, adjacent redundant detectors are switched on in place of the defective primary detectors causing the active

detector location to be shifted in the along-track direction. This shift is in addition to the normal even/odd detector offset. For TIRS, there is no even/odd offset and instead of having individual redundant detectors that must be switched on individually, an entire redundant row of detectors is downlinked for each band. Detector replacement is performed in Level 0 processing where the samples from defective primary detectors are swapped with the samples from the corresponding detector in the redundant row. The net result is that TIRS detectors will have an integer detector offset of either 0, for detectors from the primary row, or whatever the line offset happens to be between the primary and redundant detector rows, for detectors that are replaced/deselected. Since the selection of primary and redundant rows will be made separately for each band on each SCA, these offsets can vary from SCA-to-SCA but will be constant within a given band on a given SCA.

7.3.2 TIRS Line-of-Sight Projection/Grid Generation

7.3.2.1 Background/Introduction

The line-of-sight (LOS) projection and grid generation algorithm uses the TIRS LOS model, created by the TIRS LOS model creation algorithm, to calculate the intersection of the projected lines-of-sight from selected TIRS detector samples (pixels) with an Earth model (WGS84). The spacecraft position and pointing, TIRS instrument alignment and offset information, TIRS scene select mirror (SSM) angle, and image timing data contained in the LOS model are used to construct the LOS for an individual TIRS detector at a particular sample time. We then calculate the location where that line of sight intersects the Earth's surface, as defined by the WGS84 Earth ellipsoid or a specified elevation above or below that ellipsoid. LOS intersections for an array of detector samples that span each TIRS SCA and spectral band are computed at the WGS84 ellipsoid surface as well as at a range of elevation levels selected to span the actual terrain elevations found in the image area. The resulting array of projected lines-of-sight forms a three-dimensional grid of input (Level 1R) image pixel line/sample to output space (Level 1G) mappings that can be used to interpolate input/output pixel mappings for intermediate points. The resulting ability to rapidly compute input/output mappings greatly facilitates image resampling.

The TIRS LOS projection and grid generation algorithm can also work in an "inertial direction" mode in which the output space is in angular units with respect to a set of reference inertial directions. This mode is used to process lunar data wherein the inertial coordinates (declination and right ascension) of the moon, computed from a planetary ephemeris, are used as the reference to define the output image frame. In this case the lines-of-sight are computed in inertial coordinates but are not projected to the Earth's surface. This mode of operation is not specifically required for TIRS imagery, but the capability is retained in this algorithm to maintain compatibility with the corresponding OLI algorithm, to facilitate future convergence.

Concerns about the temporal (line direction) grid density that would be required to adequately capture attitude deviations (jitter) at frequencies above 10 Hz motivated the addition of new grid functionality to support high frequency image correction at image resampling time. Specifically, jitter sensitivity coefficients were added to each grid cell to allow the high frequency attitude data in the TIRS line-of-sight model jitter table to be converted to corresponding input image space line/sample offsets. These coefficients are used by the resampler to compute high frequency line/sample corrections that refine the output-to-input space image coordinate mappings provided by the grid. This allows the grid to model only lower frequency effects making a sparser grid sampling in the time (line) direction possible.

Due to layout of the TIRS focal plane, there is an along-track offset between the spectral bands within each SCA, an along-track offset between the outboard (odd) and inboard (even) SCAs, and a reversal of the band ordering in adjacent SCAs. This leads to an along-track offset in the imagery coverage area for a given band between odd and even SCAs as well as an offset between bands within each SCA. To create more uniform image coverage within a geometrically corrected output product, the leading and trailing imagery associated with these offsets is trimmed (at image resampling time) based on image active area bounds stored in the grid.

The TIRS LOS projection and grid generation algorithm is derived from the corresponding OLI algorithm. Its implementation is very similar to the oligrid application.

7.3.2.2 Dependencies

The TIRS LOS projection and grid generation algorithm assumes that the TIRS LOS model creation algorithm has been executed to construct and store the TIRS LOS model.

7.3.2.3 Inputs

The TIRS LOS projection and grid generation algorithm and its component sub-algorithms use the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs
ODL File (implementation)
CPF File Name
TIRS LOS Model File Name
DEM File Name
Reference Grid File (optional) (for transferring framing parameters)
Reference Band (optional) (reference grid band to use)
Output Image Framing Parameters:
WRS Path for path-oriented scene framing (not necessarily the LOS model path)
WRS Row for path-oriented scene framing (not necessarily the LOS model row)
Map Projection (UTM, SOM, PS)
UTM Zone (use 0 to have code compute the zone)
Map Projection Parameters
Output Pixel Size(s)
Output Image Orientation
Frame Type (e.g., MINBOX)
Frame Bounds (e.g., corner coordinates, image size)
Grid Options:
Bands to Grid
CPF file contents
Maximum detector offset for each band
Thresholds and Limits (replaces System Table)
Grid Density (line/sample/height)
Default (WGS84) Spheroid and Datum Codes
TIRS LOS Model file contents (see TIRS LOS Model Creation ADD for details)
WGS84 Earth Model Parameters
Earth Angular Velocity (rotation rate) in radians/second
Speed of light (in meters/second)
Acquisition Type (Earth, Lunar, Stellar)
TIRS to ACS reference alignment matrix
Spacecraft CM to TIRS offset in ACS reference frame (new)
SSM model (Telescope alignment matrix and time-indexed SSM angles)
Focal plane model parameters (Legendre coefs)
Detector delay table
Smoothed ephemeris at 1 second intervals (original and corrected)
Low-pass filtered attitude history (original and corrected)
High frequency attitude perturbations (roll, pitch, yaw) per image line (jitter table)
Image time codes
Integration Time
Nominal detector alignment fill table
L0R detector alignment Fill Table
DEM file contents

Min and Max Elevation
NOVAS Planetary Ephemeris file contents (Note: The NOVAS ephemeris file name is provided via an environment variable.)
JPL Ephemeris Table (DE405) for celestial bodies (i.e., the moon) (see note 1)

7.3.2.4 Outputs

TIRS Grid (see Tables 1 and 2 below for detailed grid structure contents)
Grid Header (WRS path/row, acquisition type)
Output Image Framing Information (corner coordinates, map projection)
Image active area latitude/longitude bounds (for each band)
Grid Structure Information (number of bands/SCAs)
Grid Structures (one per SCA, per band)
Band number
Image dimensions (line/sample)
Pixel size
Grid cell size (image lines/samples per cell)
Grid dimensions (# rows/# columns/# Z-planes)
Z-plane zero reference and height increment
Arrays of input line/sample grid point coordinates
Arrays of output line and sample grid point mappings
Arrays of even/odd offset coefficients (2 per grid cell)
Arrays of forward (input/output) mapping polynomials (8 per grid cell per Z-plane)
Arrays of inverse (output/input) mapping polynomials (8 per grid cell per Z-plane)
Arrays of roll-pitch-yaw jitter line sensitivity coefficients (3 per grid cell per Z-plane)
Arrays of roll-pitch-yaw jitter sample sensitivity coefficients (3 per grid cell per Z-plane)
Rough mapping polynomials (one set per Z-plane)

7.3.2.5 Options

A NOVAS planetary ephemeris file (JPL DE405) must be provided when the Acquisition Type (in the LOS model) is Lunar. The DE405 file path is provide in the JPLDE405 environment variable.

7.3.2.6 Prototype Code

Input to the executable is an ODL file; output is a HDF4 formatted resampling grid file.

The prototype code was compiled with the following options when creating the test data files:

```
-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2
```

The following text is a brief description of the main set of modules used within the prototype with each module listed along with a very short description. It should be noted that not all library modules are referenced in the explanations below. The modules within the main oligrid directory of the prototype are discussed and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

oligrid
Main driver for generating the resampling grid. Calls modules to retrieve user parameters, establish the output image frame extent, and populate the grid structure with appropriate input to output, and output to input, mapping parameters.

get_parms
This routine opens the input ODL parameter file, reads the grid parameters, closes the parameter file, and returns the parameters. Also will read the DEM, if the DEM is given as an input parameter, and determine the elevation extent within the DEM file. This elevation extent will then be used for establishing the z-plane parameters within the grid structure.

oli_get_model
Reads the OLI geometric model file and populates data within the OLI geometric model structure.

read_num_ls_l0ra
This routine extracts the number of image lines from the Level 1R image and the number of samples per band per SCA from the sensor model portion of the LOS model. The routine then returns the number of lines and samples for the input band numbers. These values, along with the grid cell size, will be used to determine grid point locations. The number of lines and samples will be returned in their respective arrays, in band-referenced order. This is similar to the manner in which the grid is stored. Thus the nlines and nsamps arrays must be of size nbands.

det_num_grid_ls
This routine will determine the number of input points to be stored in the grid according to the grid sampling rate or grid cell size chosen.

validate_utm_zone
This routine validates the UTM zone that was entered as an ODL parameter. The scene center longitude will be used for this verification. The nominal UTM zone to use is computed from the scene center longitude but the projection may be forced to an adjacent zone using input parameters. In particular, each WRS path/row may be preassigned to a UTM zone so that the same zone is always used for scenes near UTM zone boundaries. This should not introduce a zone offset greater than 1. The validation is performed by computing the UTM zone in which the scene center falls and then determining whether the input UTM zone (if any) is within one zone of the nominal zone.

oli_malloc_grid
Allocates memory for the grid based on image size and output elevation extent.

setup_jpl_solarsystem
Initializes JPL routines needed to determine position of the moon. Only used for lunar acquisitions.

calc_active_area
This routine determines the bounds of that portion of the output image frame that contains actual OLI imagery, excluding "ragged" band/SCA edges. The resulting active area bounds for each spectral band are stored in the grid for subsequent use by the image resampling logic.

north_up
This routine will determine the frame in output space for the north-up product. The actual frame is based on the output band's pixel size, but the frame is the same for every band. The method used to determine the scene corners depends on whether the corners were user input (PROJBOX) or calculated by projecting the Level 1R image corners (MAXBOX) but the framing logic is essentially the same in each case. Once given as input, or computed, the latitude/longitude scene corners are converted to the defined map projection, the extreme X and Y coordinates are found, and these extreme points are rounded to a whole multiple of the pixel size.

calc_stellar_size
Determines the output image extent for a stellar acquisition. Extent is based on SCA corners.

calc_lunar_size
Determines the output image extent for a lunar acquisition. Extent is based on either all of the SCA corners for all bands or only the SCA that contains the moon.

point_in_polygon
Simple point in a polygon check. Used with lunar process for determining if the moon lies within a SCA.

oli_moonpos_ls
Given a Level 1R line and sample location this module calculates the relative line of sight between the moon and satellite sensor.

oli_moonpos
Given a Julian day, this routine calculates the moon's position. Calls the JPL NOVAS libraries to determine the moon's position. Coordinates are given in terms of ECI true-of-date.

maxbox
This routine determines the frame in output space for the maxbox north-up product. Image framing is based on maximum image extent derived from SCA corners.

path_oriented
This routine will provide a path-oriented projection that is framed to a nominal WRS scene. The user specifies only the projection, pixel size, and the path and row of the scene.

det_grid_ls
Given the number of grid lines and samples that will be sampled in the input imagery, this routine calculates where each grid cell point will fall in the input Level 1R image. These grid cell points will fall at integer locations in the input imagery.

exx_mapedg
This routine calculates the minimum and maximum projection coordinates for given upper left and lower right latitude, longitude coordinates.

pad_corners
This routine pads the input corners by a defined factor of the pixel size. The x/y min and max values are input for the corner locations. These values are padded by PADVAL * the pixel size.

calc_center_and_rotation_angle
This routine will return the scene center and rotation angle for a nominal WRS scene. The WRS path and row of the input scene and the projection parameters are needed as input. Note: The WRS_Lat and WRS_Long are the Center_Lat_Long that need to be returned from this routine. The Heading angle is the WRS rotation angle, i.e., the image orientation relative to geodetic north.

calc_path_oriented_frame
Given the center point and rotation angle, this function will calculate the image corner coordinates in an SOM or UTM product. It also calculates the first-order polynomial coefficients which map output

line/sample coordinates to their corresponding output projection coordinates. This routine will determine the frame in output space for the path-oriented product. The frame is calculated for each band, but the frame must be the same for every band.

angle_to_map
This routine will convert the WRS rotation angle (from geodetic north) to a frame orientation angle in map coordinates. The orientation angle will be retained in the grid structure.

path_maxmin_box
This routine will provide a path-oriented product whose frame is large enough to contain all bands (maxbox).

calc_path_oriented_maxbox_frame
This routine calculates the path-oriented frame for the maxbox approach.

make_grid
This routine establishes the input to output mappings. It invokes make_grid_point for each point to compute the mapping, and then invokes make_grid_sensitivity for each point to compute the jitter sensitivity coefficients.

make_grid_point
Calculates the input to output space mapping for a single grid point. Calls oli_forward_model to perform input space location to output space location mappings.

make_grid_sensitivity
Calculates the roll-pitch-yaw to input space line/sample jitter sensitivity coefficients for one grid point. Calls oli_forward_model_pert while varying the spacecraft attitude, the input space line number, and input space sample number to determine the corresponding output space sensitivity. It then finds the input space offsets that provide the same effect in output space as a given attitude perturbation, yielding the input space correction needed to compensate for a unit jitter disturbance for each spacecraft axis.

oli_init_lunar_projtran
Initializes the position of the moon with respect the lunar acquisition. Needed for oli_lunar_projtran.

oli_forward_model
For a given a Level 1R line, sample, band and SCA location, propagates the forward (geometric) model to determine a latitude and longitude for the specified point.

oli_forward_model_pert
A variant of oli_forward_model that accepts an additional input roll-pitch-yaw attitude perturbation array. This perturbation is added to the spacecraft attitude interpolated from the OLI LOS model at the time corresponding to the input space line/sample point being projected. This capability is used by make_grid_sensitivity in determining the jitter sensitivity coefficients.

oli_findtime
This function finds the time into the scene given the Level 1R line, sample, and band. The input sample number is 0-relative and relative to the SCA.

oli_findlos
This function finds the line of sight vector in sensor coordinates, using the Legendre polynomial LOS model stored in the LOS model.

oli_findatt
This function computes the attitude, or roll, pitch, yaw, for a given time.

oli_findjit
This function is invoked by oli_forward_model when the input detector type parameter is set to EXACT. This is currently only used by the OLI L0Rp data simulator. This unit uses the input time to extract the high frequency attitude correction from the jitter table in the OLI LOS model, so that it can be added to the low frequency spacecraft attitude result in oli_forward_model. This unit is not invoked by grid generation processing, where the detector type is NOMINAL, but as part of the forward line-of-sight model, it is described here for completeness.

l8_movesat
This function computes the satellite position and velocity at a delta time from the ephemeris reference time using Lagrange interpolation.

l8_attitude
This function finds the line of sight vector from the spacecraft to a point on the ground by transforming the line of sight vector in sensor coordinates to perturbed spacecraft coordinates.

geo_center_mass_corr
Adjusts the observation vector according to the spacecraft center of mass.

geo_corr_vel_aberr
Adjusts line of sight vector for velocity aberration.

geo_findtarpos
This function finds the position where the line of sight vector intersects the Earth's surface. Used only for Earth based acquisitions.

geo_corr_light_travel_time
Adjusts target location according to the light travel time. Used only for Earth based acquisitions.

geo_centh2det
This function converts between geocentric and geodetic coordinates. Used only for Earth based acquisitions.

exx_cart2sph
Convert between cartesian and spherical coordinates. For grid generation, applies only towards stellar and lunar acquisitions.

exx_projtran
This function converts coordinates from one map projection to another. The transformation from geodetic coordinates to the output map projection depends on the type of projection selected. The mathematics for the forward and inverse transformations for the Universal Transverse Mercator (UTM), Polar Stereo Graphic, and the Space Oblique Mercator (SOM) map projections are handled

by U.S Geological Survey's (USGS) General Cartographic Transformation Package (GCTP), which may be obtained at http://edcftp.cr.usgs.gov/pub/software/gctpc/.

oli_lunar_projtran
Calculates the output line and sample location given the right ascension and declination angles associated with the sensor line-of-sight vector of a lunar acquisition. Serves as the equivalent exx_projtran for a lunar based acquisition.

exx_proj_err
This function reports projection transformation package errors. The function receives a GCTP error code and prints the correct error message.

gctp
Map projections are handled by U.S Geological Survey's (USGS) General Cartographic Transformation Package (GCTP), which may be obtained at http://edcftp.cr.usgs.gov/pub/software/gctpc/.

xxx_eval
Applies a polynomial at a given point.

calc_map_coefs
This routine calculates the bilinear mapping coefficients for each grid cell. Coefficients are calculated for mapping from input location to output location (forward mapping) and for mapping from output location to input location (inverse mapping). A separate mapping function is used for lines and samples. This equates to four mapping functions. A set of four mapping functions is calculated for each grid cell, for each SCA, for every band, and for every elevation plane that is stored in the grid.

exx_calc_forward_mappings
This function, given grid points in both input and output space, uses the Calculate Map Coefficients algorithm described in the Procedure section to generate the mapping polynomial coefficients needed to convert from a line/sample in input space (satellite) to one in output space (projection). It generates these coefficients for every cell in the grid.

exx_calc_inverse_mappings
This function, given grid points in both input and output space, uses the Calculate Map Coefficients algorithm described in the Procedure section to generate the mapping polynomial coefficients needed to convert from a line/sample in output space (projection) to one in input space (satellite). It generates these coefficients for every cell in the grid.

calc_rough_map_coefs
This routine will find the rough mapping coefficients for the grid.

oli_grid_cell_poly
This utility function calculates a "rough" mapping of output to input lines/samples. The coefficients returned from this function are used as a rough estimate of an inverse model.

calc_det_offsets

This function computes the detector offset values and stores linear mapping coefficients associated with detector offsets in the grid structure.

oli_all_ols2ils
This utility routine maps an output space line/sample back into its corresponding input space line/sample. This is done using the "rough" polynomial from the grid to determine an initial guess at an input space line and sample. From this initial guess a grid cell row and column is calculated and the inverse coefficients for that cell are retrieved from the grid. These coefficients are used to determine an exact input space line and sample (in extended space).

oli_findgridcell
This utility function finds the correct grid cell that contains the output line/sample location. It finds the correct grid cell containing the output pixel by first determining the set of grid cells to be checked. It then calls a routine to perform a "point in polygon" test on each of these grid cells to determine if the pixel does indeed fall within that grid cell.

7.3.2.7 Procedure

The LOS Projection algorithm uses the TIRS LOS model created by the TIRS LOS Model Creation algorithm to relate TIRS image pixels to ground locations or, in the case of lunar/stellar images, to ECI directions. The LOS model contains several components including: Earth orientation parameters, an image model (validated image time codes), a sensor model (including SSM angles), an ephemeris model, and an attitude model. The Level 1R image line/sample location is used to compute a time of observation (from the image model), a LOS vector (from the sensor model), the spacecraft position (from the ephemeris model) at the time of observation, and the spacecraft attitude (from the attitude model) at the time of observation. The LOS vector is projected to the Earth's surface, either the topographic surface at a specified elevation (e.g., derived from an input Digital Elevation Model), or the WGS84 ellipsoid surface, to compute the ground position associated with that Level 1R image location. This LOS projection procedure relating an input image location to an output ground location is referred to as the forward model. In image resampling, we typically need to find the Level 1R input space line/sample location corresponding to a particular Level 1G output space location so that the corresponding image intensity can be interpolated from the Level 1R data. This "inverse model" computation must be performed for every pixel in the output Level 1G product. To make this computation efficient, we create a table, or grid, of input/output mappings, parameterized by height, for use by the TIRS image resampling algorithm. Both the forward model and grid generation procedures are described in this algorithm description document.

The Geometric Grid
The geometric grid provides a mapping from input Level 1R line/sample space to output Level 1G line/sample space. As such, it incorporates not only the sensor LOS to Earth intersection geometry captured by the forward model, but also the output image framing information, such as scene corners, map projection, pixel size, image orientation, and the bounds of the active image area for each band. The gridding procedure generates a mapping grid that defines a transformation from the instrument perspective (input space) to a user specified output projection on the ground (output space). This output frame may be map-oriented (north-up) or path-oriented for Earth-view acquisitions. Alternatively, the user may specify a previously generated grid file such that this grid’s scene framing information is used for the generation of the new grid. Celestial (lunar/stellar) acquisitions use an output frame based on inertial right ascension and declination coordinates. Once the frame is determined in output space, the input space is gridded. Then the grid in input space is mapped to the output space using the forward model. Transformation coefficients to transform a grid cell from input

to output space are determined, as well as coefficients to transform a grid cell from output to input space.

The concept behind creating this resampling grid is to define only a sparse set of points for the relationship between an input line and sample location to output line and sample location (see Figure 1). Four grid points define a grid cell. A grid cell is defined as a rectangle in input space but will be distorted when mapped to the output space. The sampling of points between grid cell points is chosen such that any two points defining a grid cell and a line in input space will map to a line in output space. Therefore every grid cell defines a bilinear mapping between the input and output space and vice versa. The method of only mapping and storing a small set of input points is much more efficient than trying to map points individually by invoking the LOS model for each point. This is especially the case since a rigorous implementation of the inverse model would have to be iterative.

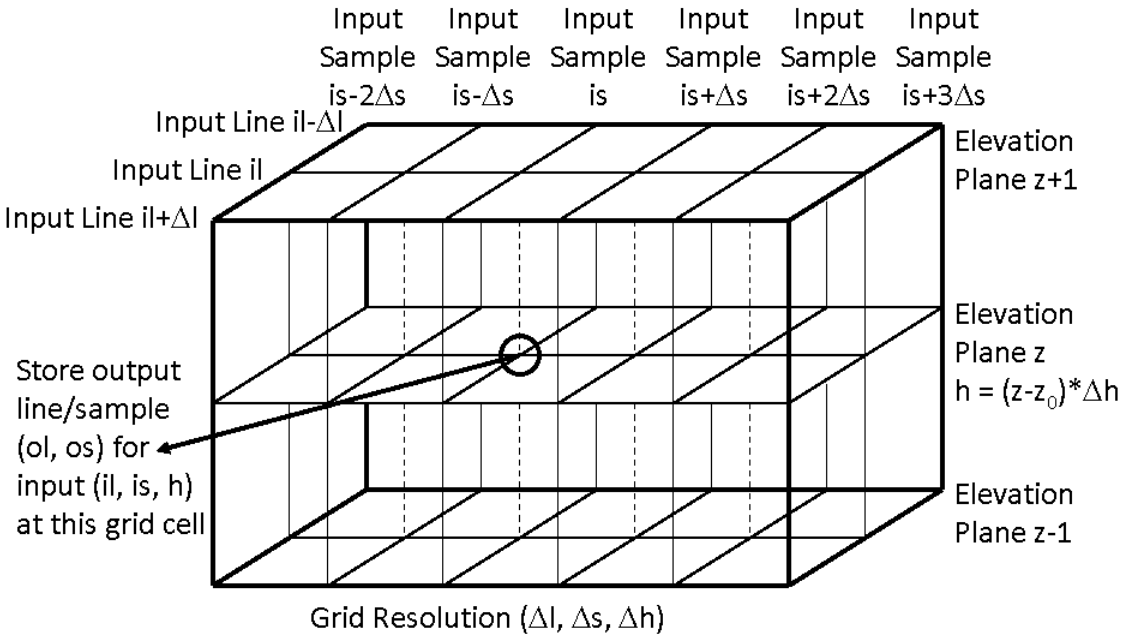


Figure 1: The 3D grid structure stores the output space line/sample coordinates corresponding to an array of input space line/sample/height coordinates.

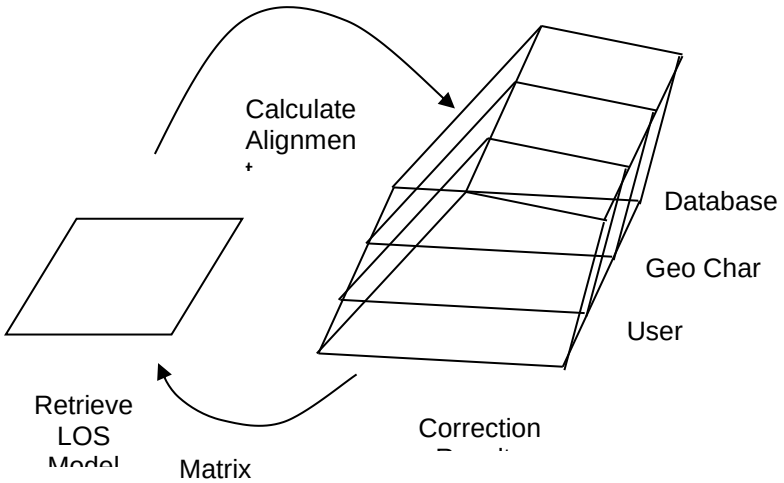


Figure 2: Forward and Inverse Mapping Using the Grid

The LOS projection grid contains projection information and three groups of mapping coefficients—one for mapping each grid cell from output space to input space (inverse), a second for mapping each grid cell from input space to output space (forward), and third that gives an approximation or "rough" mapping of output space to input space. The first two mappings are described by a set of bilinear polynomials. The input space is represented by a line and sample location while the output space is represented by a line and sample location along with a Z component, where Z represents elevation. The output lines and samples can in turn be converted to X, Y projection space location by using the output image's upper left projection coordinate and pixel size information in the grid header. Figure 2 shows how one input grid cell is mapped to a number of output grid cells, each grid cell representing a different elevation.

The number of grid cells is dependent on the line and sample size of each grid cell in the input image, elevation maximum, elevation minimum, and elevation increment. The input space is made up of evenly spaced samples and lines, values are associated with integer locations and can be indexed by an array of values: input_line[row] and input_sample[column]. Row refers to the index number, or row number, associated with the line spacing while column refers to the index number, or column number, associated with the sample spacing. The output lines and samples typically do not fall on integer values (see Figure 3). This creates a two dimensional array of indices for output line and sample locations. Adding elevation indices produces a three dimensional array for output line and sample locations. The output lines and samples are then indexed by output_line[z][row][column] and output_sample[z][row][column] where Z refers to an elevation value. The row and column are the indices associated with the gridding of the raw input space. Since there is a mapping polynomial for each grid cell, the mapping polynomial coefficients are indexed by the same method as that used for output lines and samples; i.e. there are z*row*column sets of mapping coefficients.

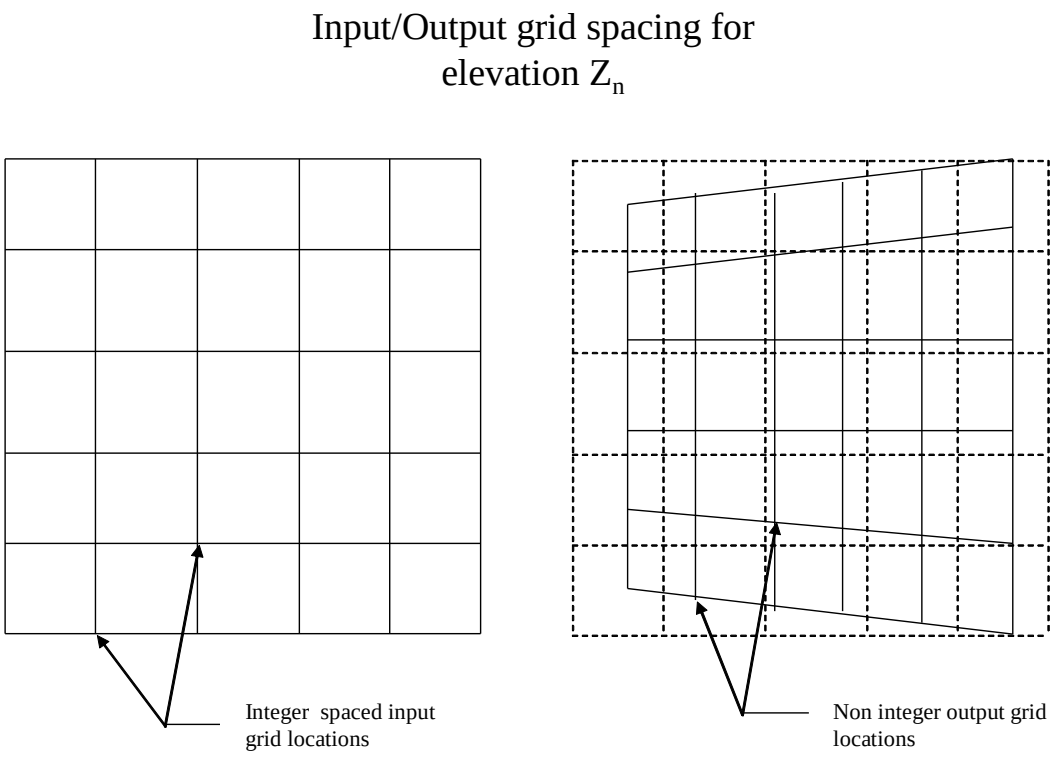


Figure 3: Mapping integer locations to "non-integer" locations

If a grid is being generated for a non-terrain corrected image (i.e., no correction for relief is being applied) then the index for z is set such that $z_{elev=0}$ = zero elevation. Note that $z_{elev=0}$ does not necessarily have to be the first index in the array since there could be values for negative elevations. If the grid is being generated for a terrain corrected image, then the indexes z_n and z_{n+1} are used such that the elevation belonging to the output location falls between the elevations associated with the indexes n and n+1. When performing an inverse mapping for a terrain corrected image, two sets of input lines and samples are calculated from the polynomials for n and n+1. The actual input line and sample is interpolated between these lines and samples.

Example:

Output line/sample has r = row, c = col and z=n, n+1. If the inverse mapping coefficients are *a* and *b* for line and sample respectively then:

$input\_line_n$ = *bilinear*(a_n,output_line,output_sample)
$input\_sample_n$ = *bilinear*(b_n,output_line,output_sample)
$input\_line_{n+1}$ = *bilinear*(a_{n+1},output_line,output_sample)
$input\_sample_{n+1}$ = *bilinear*(b_{n+1},output_line,output_sample)

bilinear is the bilinear mapping function (described below) for each grid cell.

If *e* is the elevation for the output line and sample location then the weights used to interpolate between the two input line/sample locations are:

$$w_n = \frac{e_{n+1} - e}{e_{n+1} - e_n} \qquad w_{n+1} = \frac{e - e_n}{e_{n+1} - e_n}$$

e_n, e_{n+1} and *e* are the elevations associated with z_n , z_{n+1} , and the output line and sample respectively.

The final line/sample location is found from:

$input\_line = w_n * input\_line_n + w_{n+1} * input\_line_{n+1}$
$input\_sample = w_n * input\_sample_n + w_{n+1} * input\_sample_{n+1}$

The grid must contain a zero elevation plane. If the input minimum elevation is greater than zero it is set to zero. If the input maximum elevation is less than zero it is set to zero.

Given the elevation maximum, minimum, and increment determine the number of z planes and the index of the zero elevation plane. Adjust the minimum and maximum elevations to be consistent with the elevation increment.

The number of z planes is determined from:

$$\text{number of z planes} = (\text{int})\left(\text{ceil}\left(\frac{\text{elevation maximum}}{\text{elevation increment}} \right) - \text{floor}\left(\frac{\text{elevation minimum}}{\text{elevation increment}} \right) \right) + 1$$

The subsequent grid interpolation logic assumes that there are at least 2 z planes so the number of z planes is set to 2 if the calculation above results in fewer than 2 planes. This can only happen if the minimum and maximum elevations are both zero.

The plane for an elevation of zero is then found at:

$$z_{elev=0} = -\text{floor}\left(\frac{\text{elevation minimum}}{\text{elevation increment}} \right)$$

The new minimum and maximum elevation due to the values calculated above are:

$$\text{elevation minimum} = -z_{elev=0} * (\text{elevation increment})$$
$$\text{elevation maximum} = (\text{number of z} - 1 - z_{elev=0}) * \text{elevation increment}$$

LOS Projection/Grid Generation Procedure Overview

The LOS Projection/Grid Generation procedure is executed in five stages:

6. Data Input - First, the required inputs are loaded. This includes reading the processing parameters from the input ODL parameter file, loading the TIRS LOS model from its HDF file, reading static gridding parameters from the CPF, and loading the elevation data from the DEM.
7. Scene Framing - The parameters of the output image space are computed based on the scene framing scheme specified in the input ODL file or are loaded from a previously generated grid file. This includes calculating bounds for the active image area that excludes the leading and trailing SCA imagery, and using one of several available methods for determining the Level 1G scene corners. The scene framing parameters are stored in the grid structure for eventual inclusion in the geometric metadata for the Level 1G product.
8. Grid Definition - The grid parameters are established to ensure adequate density in the space (sample), time (line), and elevation (z-plane) dimensions. The required data structures are allocated and initialized.
9. Grid Construction - The forward model is invoked for each grid intersection to construct the array of input space to output space mappings. A separate grid structure is created for each SCA and each band. The grid mapping polynomial coefficients are computed from the input space to output space mapping results for each grid cell. Once the basic grid mappings are defined, the forward model is invoked with small attitude perturbations about each axis in order to evaluate the sensitivity of the input space to output space mapping to small attitude deviations. The resulting sensitivity coefficients are stored with each grid cell for subsequent use in computing high frequency jitter corrections during image resampling. Figure 4 shows a data flow for the creation and use of these new coefficients.
10. Finalize and Output Grid - Derived grid parameters such as the global rough mapping coefficients, are added to the grid structure, and the entire structure is written to a disk file. This also includes evaluating the small, but significant, parallax effects caused by the time delay between adjacent primary and (replaced) redundant detectors as they sample the same along-track location. These effects are modeled in the grid as along- and across-track sensitivity coefficients that are scaled by the output point elevation and the even/odd detector offset, which can vary by pixel for TIRS (due to detector deselect/replacement). This parallax effect is not as pervasive in TIRS as compared to OLI since the primary TIRS detectors are not

arranged with even/odd detector stagger as is the case for OLI. The parallax correction is retained in the TIRS grid to account for bad detector replacement (when a primary row detector is replaced by the corresponding redundant row detector) and to maintain compatibility with the OLI grid.

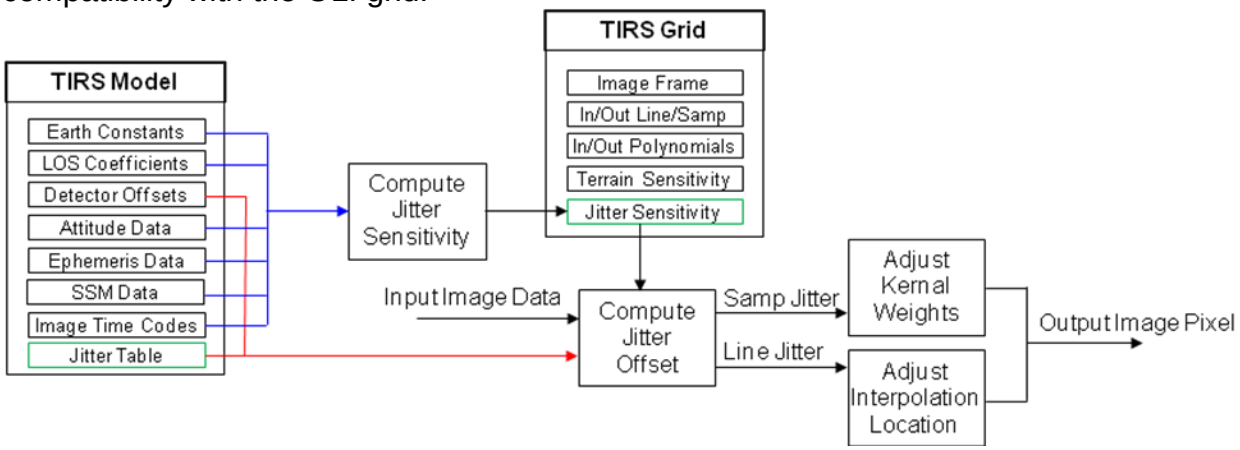


Figure 4: Jitter Correction Data Flow

Figure 5 shows a block diagram for the TIRS LOS Projection algorithm.

Stage 1 - Data Input

The data input stage involves loading the information required to perform grid processing. This includes reading the framing parameters for the output scene from the ODL file, reading grid structural parameters from the CPF, loading the TIRS LOS model structure in preparation for invoking the forward model, and reading the DEM to determine the elevation range for the image.

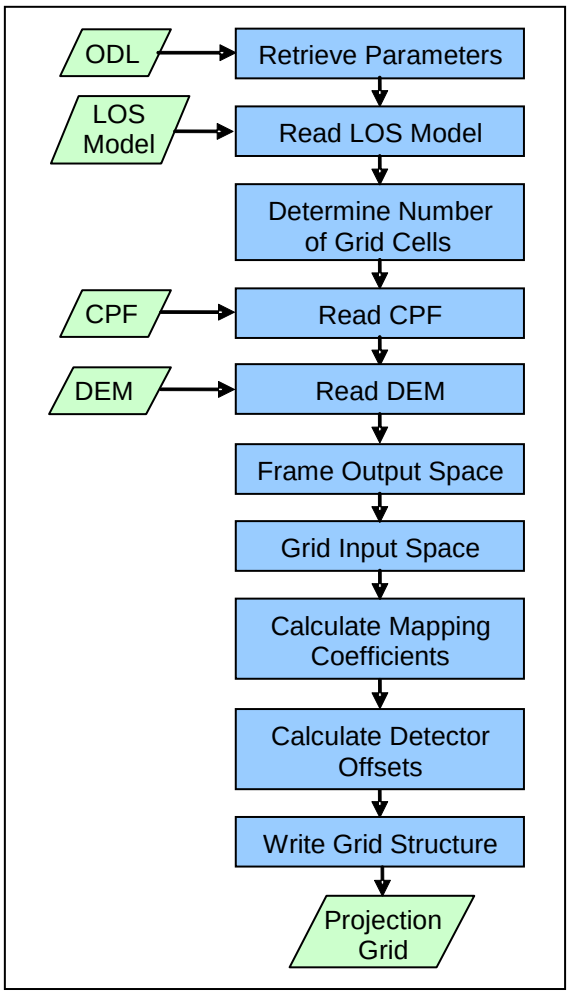


Figure 5: Line-of-Sight Projection Block Diagram

Stage 2 - Scene Framing

Framing the output image space involves determining the geographic extent of the output image to be generated by the resampler. This geographic extent of the output image space is referred to as the output space "frame," and is specified in output image projection coordinates. There are five different methods that are used to determine the output frame for Earth-viewing acquisitions. Note that the fifth method is new for TIRS and is not really a "framing method" in the same sense. Logic that supports scene framing for celestial (lunar and stellar) scenes using a maximum bounding rectangle (maxbox) approach based on inertial LOS declination and right ascension coordinates, is retained in the code reused from the corresponding OLI algorithm, but this capability is not required for TIRS so it is only described briefly below. These methods use the calculated coverage bounds of each band/SCA in different ways, with some excluding the leading and trailing SCA imagery based on a calculated active image area, and some including the leading/trailing imagery so as to preserve all available input pixels (e.g., for calibration purposes). Thus, the calculation of the active image area for each band is the first step in scene framing.

Calculating the Active Image Area

The along-track offsets between spectral bands and even/odd SCAs create an uneven coverage pattern when projected into output image space. In order to provide a more regular output image

coverage boundary, we define a rectangular active image area that excludes the excess trailing imagery from even SCAs and the excess leading imagery from odd SCAs. This active area is used for the minbox framing methods which seek to limit the output product area to provide consistent, contiguous coverage, but are ignored for maxbox framing methods, where all available imagery is desired.

The active image area is computed by constructing 8 critical SCA corner points, labeled C1 through C8 in Figure 6 below. This figure depicts the current understanding of the TIRS field of view orientation with respect to object space, but the algorithm described here will work so long as the SCAs are numbered sequentially across the field of view, in either direction. Points C1 and C2 define the top edge of the active area, C3 and C4 the right edge, C5 and C6 the bottom edge, and C7 and C8 the left edge. Note that points C4 and C5 are the same (the lower right corner of SCA01) as are points C6 and C7 (the lower left corner of SCA03). The forward model projects these 8 line/sample locations to object space, computing the latitude/longitude coordinates of the WGS84 ellipsoid intersection for each point.

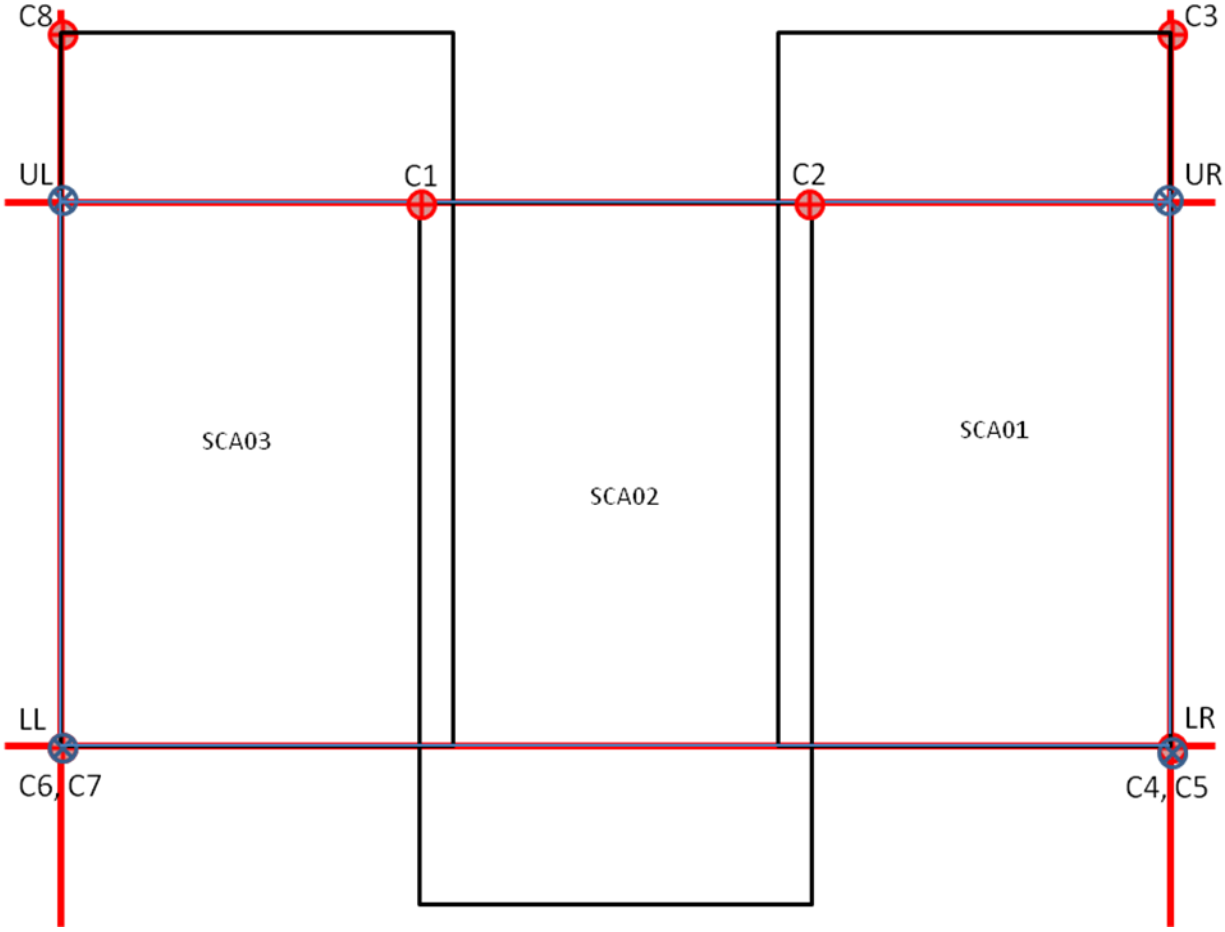


Figure 6: Active Image Area Construction

The corner point assignments are made automatically by examining the SCA across-track and along-track Legendre coefficients to determine: 1) whether SCA01 is on the left (+Y) or right (-Y) side of the scene; 2) whether even or odd SCAs lead; and 3) whether the sample number increases in the –Y or +Y direction. If the across-track Legendre constant term (coef_y0) for SCA01 is positive then it is the left-most SCA and SCA03 is the right-most. If the along-track Legendre constant term (coef_x0) for

SCA01 is greater than that for SCA02, then the odd SCAs lead. If the across-track Legendre linear term (coef_y1) for SCA01 is negative, then the sample number increases in the –Y direction.

Having determined the orientation of the SCAs, we assign the top edge to the left-most leading SCA upper left (UL) corner and the right-most leading SCA upper right (UR) corner, the right edge to the right-most SCA UR and lower right (LR) corners, the bottom edge to the right-most trailing SCA LR corner and left-most trailing SCA lower left (LL) corner, and the left edge to the left-most SCA LL and UL corners. As shown in the figure, for the TIRS: C1 = SCA02 (left-most leading SCA) UL, C2 = SCA02 (right-most leading SCA) UR, C3 = SCA01 (right-most SCA) UR, C4 = SCA01 (right-most SCA) LR, C5 = SCA01 (right-most trailing SCA) LR, C6 = SCA03 (left-most trailing SCA) LL, C7 = SCA03 (left-most SCA) LL, and C8 = SCA03 (left-most SCA) UL. Note that these assignments are based on the current TIRS SCA ordering of SCA-B = SCA01, SCA-C = SCA02, and SCA-A = SCA03, and could change if the SCA numbering system is revised. If this were to happen, the change would be reflected in the Legendre coefficients, so the logic described here would automatically compensate.

The geodetic latitudes computed by the forward model are converted to geocentric longitudes using:

$\theta = \arctan((1-e^2) \tan(\phi))$

where: θ = geocentric latitude

ϕ = geodetic latitude

e^2 = WGS84 ellipsoid eccentricity squared

This creates a set of 8 geocentric latitude/longitude (θ_i, λ_i) pairs, one for each “critical” corner, noting that geocentric longitude is equal to geodetic longitude.

Use the geocentric latitude/longitude to construct a geocentric unit vector for each corner:

$$X_i = \begin{bmatrix} \cos(\lambda_i)\cos(\theta_i) \\ \sin(\lambda_i)\cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix}$$

Note that these vectors are inherently normalized.

Construct vectors normal to the top, right, bottom, and left edge great circles by taking cross products of the corner vectors:

$$X_T = \frac{X_1 \times X_2}{|X_1 \times X_2|} \qquad X_R = \frac{X_3 \times X_4}{|X_3 \times X_4|} \qquad X_B = \frac{X_5 \times X_6}{|X_5 \times X_6|} \qquad X_L = \frac{X_7 \times X_8}{|X_7 \times X_8|}$$

Construct corner vectors from the edge vectors:

$$X_{UL} = \frac{X_T \times X_L}{|X_T \times X_L|} \qquad X_{UR} = \frac{X_R \times X_T}{|X_R \times X_T|} \qquad X_{LL} = \frac{X_L \times X_B}{|X_L \times X_B|} \qquad X_{LR} = \frac{X_B \times X_R}{|X_B \times X_R|}$$

The top and bottom edges are next checked against all of the SCA corners to ensure that any curvature in the SCA field angle pattern is accounted for. This is done to suppress residual SCA edge “raggedness”.

Adjust the top edge:

Construct a vector in the plane of the top edge great circle:

$$X_g = \frac{(X_{UR} - X_{UL}) \times X_T}{|(X_{UR} - X_{UL}) \times X_T|}$$

Initialize the minimum "out of plane" distance: $a_{min} = 1$
For each SCA:
For the two upper corners: UL (0,0) and UR (ns-1,0):
Use the forward model to project the corner.
Convert the geodetic latitude to geocentric latitude as above.
Construct a geocentric unit vector, X_i, as above.
Project the unit vector onto the X_g and X_T vectors and compute the ratio:

$$a_i = \frac{X_i \bullet X_T}{X_i \bullet X_g}$$

If $a_i < a_{min}$
$a_{min} = a_i$
$X_{min} = X_i$
Next corner
Next SCA

If $a_{min} < 0$ then the innermost corner lies inside the current active area and we need to adjust the top edge:

$$X'_g = \frac{(X_{\min} \bullet X_T)X_T + (X_{\min} \bullet X_g)X_g}{|(X_{\min} \bullet X_T)X_T + (X_{\min} \bullet X_g)X_g|}$$

$$X'_T = \frac{X'_g \times (X_{UR} - X_{UL})}{|X'_g \times (X_{UR} - X_{UL})|}$$

And update the top corner vectors using the adjusted edge vectors:

$$X_{UL} = \frac{X'_T \times X_L}{|X'_T \times X_L|} \qquad X_{UR} = \frac{X_R \times X'_T}{|X_R \times X'_T|}$$

Adjust the bottom edge:
Construct a vector in the plane of the bottom edge great circle:

$$X_g = \frac{(X_{LL} - X_{LR}) \times X_B}{|(X_{LL} - X_{LR}) \times X_B|}$$

Initialize the minimum "out of plane" distance: $a_{min} = 1$
For each SCA:
For the two lower corners: LL (0,nl-1) and LR (ns-1,nl-1):
Use the forward model to project the corner.
Convert the geodetic latitude to geocentric latitude as above.
Construct a geocentric unit vector, X_i, as above.

Project the unit vector onto the X_g and X_B vectors and compute the ratio:

$$a_i = \frac{X_i \bullet X_B}{X_i \bullet X_g}$$

If $a_i < a_{min}$

$a_{min} = a_i$

$X_{min} = X_i$

Next corner

Next SCA

If $a_{min} < 0$ then the innermost corner lies inside the current active area and we need to adjust the bottom edge:

$$X'_g = \frac{(X_{\min} \bullet X_B)X_B + (X_{\min} \bullet X_g)X_g}{\left|(X_{\min} \bullet X_B)X_B + (X_{\min} \bullet X_g)X_g\right|}$$

$$X'_B = \frac{X'_g \times (X_{LL} - X_{LR})}{\left|X'_g \times (X_{LL} - X_{LR})\right|}$$

And update the bottom corner vectors using the adjusted edge vectors:

$$X_{LL} = \frac{X_L \times X'_B}{\left|X_L \times X'_B\right|} \qquad X_{LR} = \frac{X'_B \times X_R}{\left|X'_B \times X_R\right|}$$

Convert the four corner vectors to the corresponding geodetic latitude/longitude:

λ = atan2(X.y, X.x)

θ = atan2(X.z, $\sqrt{X.x^2 + X.y^2}$)

ϕ = atan(tan(θ) / (1-e^2))

The four latitude/longitude corners are the bounds of the active image area.

Once the active image area bounds are calculated, the output product frame is determined using one of the following methods:

Method 1: PROJBOX

The user defines the upper-left and lower-right corner coordinates of the area of interest in target map projection coordinates. These coordinates are then projected to the output projection coordinate system using the Projection Transformation Package (see the Projection Transformation sub-algorithm below). This usually results in a non-rectangular area so a minimum-bounding rectangle is found (in terms of minimum and maximum X and Y projection coordinates) in the resulting output space. This minimum-bounding rectangle defines the output space frame. The output image pixel size is then applied to the projection space to determine the number of lines and samples in the output space. This creates an output image that is map projection north-up.

Method 2: MINBOX

The image active areas for each band, calculated previously, are converted to the specified output map projection coordinate system and used in a minimum bounding rectangle computation to create an output image frame that includes the active area for each band. The

computed (latitude/longitude) active area corners are maintained in the grid for subsequent use by the image resampler, so that the output product image will not include leading/trailing SCA imagery.

Method 3: MAXBOX

The four corners of each SCA in each band are projected to the Earth. The maximum and minimum latitude and longitude found across all SCAs and all bands are used to establish the output scene frame in the manner described above for the PROJBOX method. This creates an output frame that contains all input pixels from all bands. The previously calculated image active areas are ignored in this process, and the band active area corners are all set equal to the output product corners. Leading and trailing SCA imagery is thereby not excluded from MAXBOX framed products.

Method 4: PATH

The user specifies a path oriented Landsat product in either the SOM or UTM projection. In this case, the framing coordinates are not user-specified. For a standard PATH scene, the frame is a preset number of lines and samples based on the Landsat WRS scene size and the maximum rotation needed to create a path-oriented product. For PATH_MAXBOX scenes the MAXBOX logic described above is applied to the path-oriented scene to ensure that the output frame contains all input pixels from all bands. For PATH_MINBOX the MINBOX logic described above is applied to the path-oriented scene so that the image active areas control the bounds of the path-oriented frame.

Method 4: TRANSFER

The output image framing information is transferred from an input reference grid file. This will be the primary method used for framing TIRS images as the product frame will be computed based on the OLI image footprint and then transferred to the TIRS grid. This method would be used if the optional reference grid and reference band parameters are provided as inputs. Note further that this framing method would not be required in the operational implementation if the OLI and TIRS bands are combined in a common grid structure with a common output frame.

The scene framing logic uses the following sub-algorithms/routines:

a) Validate UTM Zone

This routine validates the UTM zone that was entered as an ODL parameter. The scene center longitude will be used for this verification. The nominal UTM zone to use is computed from the scene center longitude but the projection may be forced to an adjacent zone using input parameters. In particular, each WRS path/row may be preassigned to a UTM zone so that the same zone is always used for scenes near UTM zone boundaries. This should not introduce a zone offset greater than 1. The validation is performed by computing the UTM zone in which the scene center falls and then determining whether the input UTM zone (if any) is within one zone of the nominal zone.

Shift the scene center longitude to put it in the range 0-360 degrees:
SC_long = mod(SC_long + 540, 360)

where: SC_long is the scene center longitude in degrees

Compute the nominal UTM zone (note that UTM zones are six degrees wide):
SC_zone = (int)floor(SC_long/6) + 1

See if the input zone is within one zone of the nominal zone:
if (abs(input_zone - SC_zone) < 2 or (60 - abs(input_zone - SC_zone)) < 2)
then input_zone is valid.

b) North Up Framing

This routine will determine the frame in output space for the north-up product. The actual frame is based on the optimal band's pixel size, but the frame is the same for every band. The method used to determine the scene corners depends on whether the corners were user input (PROJBOX) or calculated by projecting the Level 1R image corners (MINBOX, MAXBOX) but the framing logic is essentially the same in each case. Once input or computed, the latitude/longitude scene corners are converted to the defined map projection, the extreme X and Y coordinates are found, and these extreme points are rounded to a whole multiple of the pixel size. The north-up framing methods are each described in the following sub-algorithms.

b.1) Map Edge/PROJBOX Framing

Calculates the minimum and maximum projection coordinates for given upper left and lower right latitude, longitude coordinates.

- Calculate min/max coordinates along east edge of output area by computing latitude/longitude to map x/y projections for a series of points from (minimum latitude, maximum longitude) to (maximum latitude, maximum longitude).
- Calculate min/max coordinates along west edge of output area by computing latitude/longitude to map x/y projections for a series of points from (minimum latitude, minimum longitude) to (maximum latitude, minimum longitude).
- Calculate min/max coordinates along south edge of output area by computing latitude/longitude to map x/y projections for a series of points from (minimum latitude, minimum longitude) to (minimum latitude, maximum longitude).
- Calculate min/max coordinates along north edge of output area by computing latitude/longitude to map x/y projections for a series of points from (maximum latitude, minimum longitude) to (maximum latitude, maximum longitude).

Note that since lines of constant latitude and/or longitude may be curved in map projection space, the extreme map x/y points may not correspond to the four PROJBOX corners.

b.2) Minbox/Maxbox Framing Determine the frame in output space for the minbox or maxbox north-up product. The actual frame is determined based on the optimal band's pixel size, but the frame is the same for every band.

b).2.1 Minbox Framing Calculate the MINBOX frame bounds using the active area corner points for each band.

6. Call projtran (see below) to get the output map projected x/y, for each active area corner point for each image band.

7. Find the minimum and maximum output proj x/y from the full set of active area corner points.

8. Pad the min and max output projection x/y to make them a multiple of pixsize.

9. Fill in the corners for the grid in the order of UL, LL, UR, LR and Y/X coords.
   UL = min x, max y
   UR = max x, max y
   LL = min x, min y
   LR = max x, min y

10. Find the number of lines and samples for the grid, for each specified band number.
    lines = (max y - min y)/pixsize + 1
    samples = (max x - min x)/pixsize + 1

b).2.2 Maxbox Framing Calculate the MAXBOX product frame bounds using the projected corners of each band/SCA.

1. Find the four image corners in input space for each SCA and band.
   UL - (1, first_pixel)
   UR - (1, last_pixel)
   LL - (NLines, first_pixel)
   LR - (NLines, last_pixel)

2. Call the forward model (see below) to get the output lat/long, for each corner point.

3. Call projtran (see below) to get the output map projected x/y, for each corner point.

4. Find the minimum and maximum output proj x/y from the full set of corner points.

5. Pad the min and max output projection x/y to make them a multiple of pixsize.

6. Fill in the corners for the grid in the order of UL, LL, UR, LR and Y/X coords.
   UL = min x, max y
   UR = max x, max y
   LL = min x, min y
   LR = max x, min y

7. Find the number of lines and samples for the grid, for each specified band number.
   lines = (max y - min y)/pixsize + 1
   samples = (max x - min x)/pixsize + 1

8. Call projtran to convert the map projection Y/X coordinates of the output product corners to latitude/longitude.

9. Replace the active area corner coordinates for each band with the converted output product corner coordinates.

b.2.3) Pad Corners Pad the input corners by a defined factor of the pixel size. The x/y min and max values are input for the corner locations. These values are padded by PADVAL * the pixel size. The newly padded x/y min and max values are returned, replacing the original values.

ixmin = int (Xmin/(PADVAL*pixsize))
Xmin = ixmin*PADVAL*pixsize
ixmax = int (Xmax/(PADVAL*pixsize))+1
Xmax = ixmax*PADVAL*pixsize
iymin = int (Ymin/(PADVAL*pixsize))
Ymin = iymin*PADVAL*pixsize
iymax = int (Ymax/(PADVAL*pixsize))+1
Ymax = iymax*PADVAL*pixsize

c) Path Oriented Framing

Provide a path-oriented projection that is framed to a nominal WRS scene. The projection, pixel size, and the path and row of the scene must be defined.

c.1) Calculate Center and Rotation Angle

Calculate the scene center and rotation angle for a nominal WRS scene. The WRS path and row of the input scene and the projection parameters are needed as input. The nominal WRS scene center lat/long and rotation angle for the given projection are returned. The algorithm has the following steps:

Convert input angles to radians:
Inclination_Angle_R = Pi / 180 * Inclination_Angle
Long_Path1_Row60_R = Pi / 180 * Long_Path1_Row60

Compute the Earth's angular rotation rate:
earth_spin_rate = 2 * Pi / (24 * 3600)

Note: We use the solar rotation rate rather than the sidereal rate in order to account for the orbital precession which is designed to make the orbit sun synchronous. Thus, the apparent Earth angular velocity is the inertial (sidereal) angular velocity plus the precession rate which, by design, is equal to the solar angular rate.

Compute the spacecraft's angular rotation rate:
SC_Ang_Rate = 2 * Pi * WRS_Cycle_Orbits / (WRS_Cycle_Days*24*3600)

Compute the central travel angle from the descending node:
Central_Angle = (Row - Descending_Node_Row)/Scenes_Per_Orbit*2*Pi

Compute the WRS geocentric latitude:
WRS_GCLat = asin(-sin(Central_Angle) * sin(Inclination_Angle_R))

Compute the longitude of Row 60 for this Path:
Long_Origin = Long_Path1_Row60_R - (Path-1) * 2*Pi/WRS_Cycle_Orbits

Compute the WRS longitude:
Delta_Long = atan2(tan(WRS_GCLat)/tan(Inclination_Angle_R),
cos(Central_Angle)/cos(WRS_GCLat))
WRS_Long = Long_Origin - Delta_Long - Central_Angle *
Earth_Spin_Rate / SC_Ang_Rate

Make sure the longitude is in the range +/- Pi:
While (WRS_Long > Pi)
WRS_Long = WRS_Long - 2*Pi
While (WRS_Long < -Pi)
WRS_Long = WRS_Long + 2*Pi

Compute the scene heading:
Heading_Angle = atan2(cos(Inclination_Angle_R)/cos(WRS_GCLat),
-cos(Delta_Long)*sin(Inclination_Angle_R))

Convert the WRS geocentric latitude to geodetic latitude:

WRS_Lat = atan(tan(WRS_GCLat) * (Semi_Major_Axis/Semi_Minor_Axis) * (Semi_Major_Axis/Semi_Minor_Axis))

Convert angles to degrees:
WRS_Lat = WRS_Lat * 180 / Pi
WRS_Long = WRS_Long * 180 / Pi
Heading_Angle = Heading_Angle * 180 / Pi

Round WRS lat/long off to the nearest whole arc minute:
WRS_Lat = round(WRS_Lat*60) / 60
WRS_Long = round(WRS_Long*60) / 60

c.2) Calculate Path Oriented Frame

Calculate the center point and rotation angle, and the image corner coordinates in an SOM or UTM projection. Also calculate the first-order polynomial coefficients which map output line/sample coordinates to their corresponding output projection coordinates. Determine the frame in output space for the path-oriented product. Calculate the frame for each band. The frame must be the same for all bands.

c.2.1) Angle to Map

Convert the WRS rotation angle (from geodetic north) to a frame orientation angle in map coordinates. The following is an algorithm to compute this:

Convert the WRS scene center latitude/longitude to map projection x/y (X1, Y1) using the projtran routine.

Add 1 microradian (0.2 seconds) to the WRS scene center latitude and convert this point to map projection x/y (X2, Y2).

Compute the azimuth of this line in grid space as the arctangent of (X2-X1)/(Y2-Y1). This is the grid azimuth of geodetic north at the WRS scene center.

Add this angle to the WRS rotation angle to give the grid heading. A standard framed scene puts the satellite direction of flight at the bottom of the scene, so the scene orientation angle is the grid heading + or - 180 degrees. If the grid heading is <0 then subtract 180 degrees. If the grid heading is >0 then add 180 degrees. This is the scene orientation angle to use with the WRS scene center.

c.2.2) Path-oriented Minbox/Maxbox Frame

Calculate the path oriented frame that is large enough to contain all bands.

c).2.2.1. Calculate Path-oriented Minbox Frame

Calculate path-oriented frame for the minbox approach.

5. Compute the map projection coordinates of the four image active area corners for each band as described in step 1 of Minbox Framing.
6. Offset and rotate the scene corners to the path oriented frame using the WRS scene center map projection coordinates (X1, Y1) and orientation angle:
    a. X' = (X - X1) cos(angle) - (Y - Y1) sin(angle) + X1
    b. Y' = (X - X1) sin(angle) + (Y - Y1) cos(angle) + Y1
7. Compute the minbox frame as described in steps 2-4 of Minbox Framing.
8. Convert the rotated minbox corners back to the unrotated map projection coordinate system:

a. X = (X' - X1) cos(angle) + (Y' - Y1) sin(angle) + X1
b. Y = -(X' - X1) sin(angle) + (Y' - Y1) cos(angle) + Y1

c.2.2.2) Calculate Path-oriented Maxbox Frame

Calculate path-oriented frame for the maxbox approach.

1. Compute the map projection coordinates of the four image corners as described in steps 1-3 of Maxbox Framing.
2. Offset and rotate the scene corners to the path oriented frame using the WRS scene center map projection coordinates (X1, Y1) and orientation angle:
    a. X' = (X - X1) cos(angle) - (Y - Y1) sin(angle) + X1
    b. Y' = (X - X1) sin(angle) + (Y - Y1) cos(angle) + Y1
3. Compute the maxbox frame as described in steps 4-7 of Maxbox Framing.
4. Convert the rotated maxbox corners back to the unrotated map projection coordinate system:
    a. X = (X' - X1) cos(angle) + (Y' - Y1) sin(angle) + X1
    b. Y = -(X' - X1) sin(angle) + (Y' - Y1) cos(angle) + Y1
5. Call projtran to convert the map projection Y/X coordinates of the output product corners to latitude/longitude.
6. Replace the active area corner coordinates for each band with the converted output product corner coordinates.

d) Transfer Framing

Open and read the framing information from an input reference grid file. The map projection, image size, image bounds, pixel size, and other framing parameters are copied from the specified band in the reference grid instead of being computed using one of the other methods described above. This method will be used to transfer OLI scene frames to the corresponding TIRS data.

e) Celestial Acquisitions

Celestial acquisitions use the same framing logic as Earth acquisitions (namely maxbox) but the output space coordinate systems are sufficiently different to merit separate discussion. For both lunar and stellar acquisitions the output space is defined in terms of directions in inertial space, defined by the ECI J2000 right ascension and declination of the TIRS look vectors. In the case of stellar acquisitions, the output space "projection" uses the ECI J2000 right ascension and declination directly. For lunar acquisitions the output coordinate system is modified to use the LOS right ascension and declination offset from the lunar right ascension and declination at the time of observation. This creates a slowly rotating coordinate system that tracks the moon and is the reason for having a planetary ephemeris file as an input to this algorithm. These differences emerge in the forward model computations for celestial acquisitions where the LOS intersection logic used for Earth acquisitions is replaced by operations on the inertial lines-of-sight (after conversion to inertial right ascension and declination angles), with the resulting map projection x/y coordinates used in the Earth-view algorithms replaced by right ascension and declination (or delta-right ascension and delta-declination). The same maxbox framing logic applied to the x/y map projection coordinates in Earth-view acquisitions is then applied to these angular celestial coordinates.

Stage 3 - Grid Definition

The grid definition stage determines the required size of the grid, allocates the grid structure, and computes the input space (Level 1R) line/sample locations for each grid cell.

a) Determine Number of Grid Input/Output Lines/Samples

This routine will determine the number of input points to be stored in the grid according to the grid sampling rate or grid cell size chosen.

Loop through each band stored in the grid

Loop through each SCA stored in the grid.

Calculate the number of lines and samples stored in the grid according to the size of each grid cell and the size of the input image to be processed. Store the number of grid lines and samples calculated in the grid.

Calculate number of times grid cell size divides into Level 1R imagery

$$\text{number of grid lines} = \frac{\text{number of image lines}}{\text{grid cell size line direction}} + 1$$

$$\text{number grid samples} = \frac{\text{number of detectors per SCA}}{\text{grid cell size sample direction}} + 1$$

where:

number of image lines = number of lines in Level 1R (LOS model)
number of detectors per SCA = number of samples per SCA (LOS model)
grid cell size line direction = number of lines in one grid cell
grid cell size sample direction = number of samples in one grid cell

If the grid cell size in the line direction does not divide evenly into the number of lines in the Level 1R then increment the number of grid lines by one.

If the grid cell size in the sample direction does not divide evenly into the number of samples in the Level 1R then increment the number of grid samples by one.

b) Determine Grid Lines/Samples

Given the number of grid lines and samples that will be sampled in the input imagery, this routine calculates where each grid cell point will fall in the input Level 1R image. These grid cell points will fall at integer locations in the input imagery.

Loop through each band that is stored in the grid

Loop through each SCA stored in the grid

Initialize first grid cell line location to zero relative.

input line location grid cell$_0$ = 0

Loop until the grid cell line location is greater than or equal to the number of Level 1R lines, incrementing each new grid cell line location by the appropriate grid cell size in the line direction for the current band and SCA.

input line location grid cell$_n$ = input line location grid cell$_{n-1}$
+ grid cell size line direction

Set last grid cell line location to the last line in Level 1R image.

input line location grid $cell_{last}$ = number of lines in Level 1R imagery

Initialize first grid cell sample location to zero relative.

input sample location grid $cell_0$ = 0

Loop until the grid cell sample location is greater than or equal to the number of Level 1R samples, incrementing each new grid cell sample location by the appropriate grid cell size in the sample direction for the current band and SCA.

input sample location grid $cell_n$ = input sample location grid $cell_{n-1}$
+ grid cell size sample direction

Set last grid cell sample location to the last sample in Level 1R image.
input sample location grid $cell_{last}$ = number of samples in Level 1R imagery

Stage 4 - Grid Construction
Once the grid structures are created (one per SCA per band) the forward model is evaluated at every grid intersection, that is, for every Level 1R line/sample location at every elevation plane. The forward model computes the WGS84 latitude/longitude coordinates associated with each input line/sample/height point. These latitude/longitude positions are then converted to output space line/sample by projecting them to map x/y, computing the offsets (and rotation if path-oriented) from the upper-left scene corner, and scaling the offsets from meters to pixels using the pixel size.

a) Make Grid
This routine creates the geometric mapping grid in output space.

Given the number of grid lines and samples that will be sampled in the input imagery, loop on each band of each SCA, loop on number of z-planes, loop on number of input grid lines and samples calculating the corresponding output line and sample location. For each input line, sample location, and elevation, the instrument forward model function is called. This forward model function is outlined in the steps below. Additional detail on the sub-algorithms which comprise the forward model is provided in the subsection titled "Forward Model" later in this document.

The forward model uses the TIRS LOS model structure and the CPF to map an input line and sample location to an output geographic location. These are the steps that are performed whenever calculating an output geodetic latitude and longitude from an input line and sample by invoking the instrument “forward model.” The GCTP function can then be used to transform the geographic latitude and longitude to a map projection X and Y coordinate. If the output image has a “North up” orientation, then the upper left projection coordinate of the output imagery and the output pixel size can be used to transform any projection coordinate to an output line and sample location. If the map projection space is in a rotated projection space, such as having a satellite path orientation, then a transformation handling rotation is established between projection space and output pixel location. This transformation is then used in converting projection coordinates to output pixel line and sample locations.

The process listed below is performed on all bands, all elevation planes, and all SCAs present in the grid. The detector type used in the process is nominal (see the TIRS LOS Model Creation ADD for a discussion of detector types). The list explains the actions taken if a detector type other than nominal is chosen, so that it can be referenced later.

Loop on number of input grid lines.

Loop on number of input grid samples.

Read the input space (Level 1R) line/sample coordinate for this grid point.

Loop on the number of elevation planes.

Compute the height of the current elevation plane:

$$\text{height} = (z - z_{elev=0}) * \text{elevation\_increment}$$

where:
z is the index of the current z-plane and
$z_{elev=0}$ is the index of the zero elevation z-plane.

Invoke the forward model to compute the corresponding ground position latitude/longitude for this point. The general steps of the forward model are described here and are presented in more detail below.

Find Time
Find the nominal time of input sample relative to the start of the imagery. This procedure is described in the TIRS LOS Model Creation ADD and is repeated below in the Find Time sub-algorithm description.

Find TIRS LOS
Find the TIRS LOS vector for the input line/sample location using the Legendre polynomial coefficients and the scene select mirror angle as described below in the Find TIRS LOS sub-algorithm.

Find Attitude
Calculate the spacecraft attitude corresponding to the LOS, i.e. for the line/sample location, at the time computed above, using the Find Attitude sub-algorithm described below. Note that for Earth acquisitions the roll-pitch-yaw attitude sequence in the LOS model is relative to the orbital coordinate system whereas for celestial (lunar/stellar) acquisitions the LOS model roll-pitch-yaw sequence is with respect to the ECI J2000 coordinate system. The operations applied by the Find Attitude sub-algorithm are the same in either case.

Find Ephemeris
Calculate satellite position for line/sample using Lagrange interpolation. Reference the move_sat sub-algorithm described in the TIRS LOS Model Creation ADD and repeated below. Note that for Earth acquisitions the move_sat sub-algorithm is provided with the

corrected ECEF ephemeris data from the LOS model whereas for celestial (lunar/stellar) acquisitions it will be passed the corrected ECI ephemeris.

Rotate LOS to ECEF (Earth-view) or ECI (Celestial)
Use the TIRS alignment matrix in the TIRS LOS model to convert the LOS vector from sensor to ACS/body coordinates. Then apply the interpolated roll, pitch, and yaw to the LOS to convert ACS/body to orbital (Earth-view) or ECI (celestial). If Earth-view, use the ephemeris to construct the orbital to ECEF rotation matrix and use it to transform LOS to ECEF. The procedure for Earth-view scenes is described in the Attitude sub-algorithm below. For celestial acquisitions, the procedure is complete once the LOS has been rotated to ECI using the roll-pitch-yaw perturbation matrix.

Spacecraft Center of Mass to TIRS Offset Correction
Adjust the spacecraft position for the offset between the spacecraft center of mass and the TIRS instrument. This offset, in spacecraft body coordinates, is stored in the LOS model structure. First, convert the offset from spacecraft body frame to ECEF using the attitude perturbation matrix (body to orbital) and the orbital to ECEF matrix:

$$[\text{orbital CM to TIRS}] = [\text{perturbation}][\text{body CM to TIRS}]$$

$$[\text{ECEF CM to TIRS}] = [\text{ORB2ECEF}][\text{orbital CM to TIRS}]$$

Add the offset to the ECEF spacecraft position vector. This correction is not used for celestial (lunar/stellar) acquisitions.

Correct LOS for Velocity Aberration
The relativistic velocity aberration correction adjusts the computed LOS (ECEF for Earth-view and ECI for celestial) for the apparent deflection caused by the relative velocity of the platform (spacecraft) and target. The preparatory computations are somewhat different for Earth-view and celestial acquisitions due to the differences in target velocity.

Earth-view Case
The LOS intersection sub-algorithm described below (see Find Target Position) is invoked with an elevation of zero to find the approximate ground target position. The ground point velocity is then computed as:

$\mathbf{V}_g = \omega \times \mathbf{X}_g$

where:

$\mathbf{V}_g$ = ground point velocity
$\mathbf{X}_g$ = ground point ECEF position
ω = Earth rotation vector = $[\,0 \quad 0 \quad \Omega_e\,]^T$
Ω_e = Earth rotation rate in radians/second (from CPF)

The relative velocity is then:

$\mathbf{V} = \mathbf{V}_s - \mathbf{V}_g$

where Vs is the spacecraft ECEF velocity from the ephemeris data.

Correcting the Earth-View LOS
The LOS vector is adjusted based on the ratio of the relative velocity vector to the speed of light (from the CPF):

$$\boldsymbol{l}' = \frac{\boldsymbol{l} - \frac{\boldsymbol{V}}{c}}{\left| \boldsymbol{l} - \frac{\boldsymbol{V}}{c} \right|}$$

where: **l** = uncorrected LOS and **l'** = corrected LOS

Note that in this case the LOS velocity aberration correction is negative since we are correcting the apparent LOS to the true (aberration corrected) LOS. The correction is positive if we are computing the apparent LOS from the true (geometrical) LOS (see lunar case below).

Celestial (Lunar/Stellar) Case
Both lunar and stellar acquisitions use the spacecraft inertial velocity from the ephemeris data as the relative velocity. This is justified by the use of a lunar ephemeris (using the Naval Observatory's NOVAS-C package) that returns apparent places. The apparent location of the moon is already corrected for light travel time (see below) and velocity/planetary aberration due to the motion of the moon around the Earth. Thus, the residual aberration is due only to the motion of the spacecraft relative to the Earth. Thus, for both lunar and stellar acquisitions:

$\mathbf{V} = \mathbf{V}_s$

where Vs is the spacecraft ECI velocity from the ephemeris data.

Correcting the Celestial LOS
For stellar acquisitions, the LOS is corrected for aberration in the same manner as for Earth-view scenes. For lunar acquisitions, rather than correct the LOS vector, we adjust the apparent location of the moon. The lunar vector is thus adjusted based on the ratio of the relative velocity vector to the speed of light (from the CPF) as:

$$\mathbf{l}' = \frac{\mathbf{l} + \frac{\mathbf{V}}{c}}{\left| \mathbf{l} + \frac{\mathbf{V}}{c} \right|}$$

where: **l** = uncorrected LOS and **l'** = corrected LOS

The correction is positive in this case since we are computing an apparent location rather than correcting one.

LOS Intersection
For Earth-view acquisitions, intersect the LOS in ECEF with the Earth model as described in the Find Target Position sub-algorithm below. This yields the geodetic latitude, longitude, and height of the ground point.

For celestial acquisitions, convert the ECI LOS to right ascension (RA) and declination (δ) angles:

$$RA = \tan^{-1}(\frac{y}{x})$$

$$\delta = \tan^{-1}(\frac{z}{\sqrt{x^2 + y^2}})$$

where the ECI los vector is $[\ x\ \ y\ \ z\ ]^T$.

Correct Ground Position for Light Travel Time
Since the light departing the ground point takes a finite time to arrive at the TIRS sensor, there is a slight discrepancy in the corresponding time at the ground point and at the spacecraft. Since the LOS intersection logic assumed that these times were the same, a small correction can be made to correct for this light travel time delay.

Given the ECEF positions of the ground point and the spacecraft, compute the light travel time correction as follows:

Compute the distance from the ground point to the spacecraft:

$$d = \left|\mathbf{X_s} - \mathbf{X_g}\right|$$

where:

$\mathbf{X}_s$ is the spacecraft ECEF position and
$\mathbf{X}_g$ is the ground point ECEF position.

Compute the light travel time using the speed of light (from CPF):

$$ltt = \frac{d}{c}$$

Compute the Earth rotation during light travel:

$\theta = ltt * \Omega_e$ where Ω_e is the Earth angular velocity from the CPF.

Apply the light travel time Earth rotation:

$$\mathbf{X}'_g = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{X}_g$$

where:

$\mathbf{X}_g$**'** is the corrected ECEF position
X_g is the uncorrected ECEF position

Convert the corrected ECEF position to geodetic latitude, longitude and height.

Note that the light travel time correction for lunar observations due to the difference between the Earth-moon distance and the spacecraft-moon distance is neglected. This is justified by the fact that that the lunar angular rate is less than 3 microradians per second

and the maximum LTT difference is about 25 milliseconds making the magnitude of this effect less than 0.1 microradians.

Convert Position to Output Space Line/Sample

The angular geodetic (latitude/longitude) or celestial (RA/declination) coordinates must be converted to the corresponding output space line/sample coordinate to complete the input space to output space mapping.

For Earth-view acquisitions this is accomplished as follows:

Calculate the map projection X/Y for the geodetic latitude and longitude.

Convert map X/Y coordinate to output line/sample location:

If the output map projection is of a path-oriented projection then the X/Y coordinate is transformed to output space with a bilinear transformation.

$$line = a_0 + a_1 * X + a_2 * Y + a_3 * X * Y$$

$$sample = b_0 + b_1 * X + b_2 * Y + b_3 * X * Y$$

where:
a_i = polynomial coefficients that map X/Y to an output line location
b_i = polynomial coefficients that map X/Y to an output sample location
X,Y = map projection coordinates

The polynomial transformation is set up to handle the rotation involved in rotating a "Map North" projection to Satellite of "Path" projection (i.e. one that has the output line coordinate system more closely aligned with the along flight path of the satellite).

If the output map projection is not path-oriented, but "North up," the relationship between X/Y and output line/sample does not involve any rotation and the following equation is used:

$$\text{line} = \frac{\text{upper left Y} - \text{Y}}{\text{pixel size Y}}$$

$$\text{sample} = \frac{\text{X} - \text{upper left X}}{\text{pixel size X}}$$

where:
upper left Y = upper left Y projection coordinate of output image
upper left X = upper left X projection coordinate of output image
pixel size Y = output pixel size in Y coordinates
pixel size X = output pixel size in X coordinates

Note that these line and sample pixel coordinates are (0,0) relative (i.e., the center of the upper left pixel is at line,sample 0,0).

For lunar acquisitions, the right ascension and declination angles derived from the inertial LOS are offset from the nominal lunar inertial position to establish an output frame that "tracks" the apparent location of the moon. This is done as follows:

a) Compute the apparent ECI J2000 position of the moon.

5. Use the input JPL lunar ephemeris data in the NOVAS-C package to compute the ECI true-of-date (ECITOD) apparent location of the moon at the time corresponding to the current LOS (lxx_moonpos). This apparent location is provided as an ECITOD vector (i.e., including both direction and distance).
6. Apply the nutation and precession corrections (see Ancillary Data Preprocessing ADD for additional information) to convert the ECITOD vector to ECI J2000.
7. Subtract the current spacecraft ECI J2000 position vector from the lunar ECI J2000 vector to compute the spacecraft-lunar vector.
8. Compute the apparent (parallax corrected) right ascension, declination, and spacecraft-lunar distance from the spacecraft-lunar vector (by invoking exx_cart2sph.

b) Compute the differences between the LOS right ascension and declination and the apparent lunar right ascension and declination.

c) Normalize the nominal angular pixel size by the ratio of the current spacecraft-moon distance (computed above) and the nominal spacecraft-moon distance. The nominal distance is computed at the acquisition center time.

$$psize_{current} = psize_{nominal} * distance_{nominal} / distance_{current}$$

d) Divide the angular distances computed in b) above by the normalized pixel size computed in c) above. This yields the moon-relative line/sample coordinate. This is the coordinate space in which lunar images are framed, so the offset between these coordinates and the lunar scene upper left corner coordinates yields the output space line/sample for the current grid point.

For stellar acquisitions, the right ascension and declination angles derived from the inertial LOS are used directly. The offsets relative to the scene upper left corner (in right ascension/declination space) are computed and divided by the angular pixel size to compute output space line/sample coordinates.

One additional note regarding the celestial acquisition scene framing is in order. Since right ascension, like longitude, increases eastward, and declination, like latitude, increases northward, and given that celestial images are looking "up" rather than "down", the right ascension-x, declination-y coordinate system is left-handed. This can lead to the moon being apparently inverted left-to-right in the output image. This is not important for the applications (e.g., band registration characterization) in which the lunar images are to be used. If "anatomically correct" lunar images are required, some changes to the framing logic may be necessary.

The line and sample location calculated is stored in the grid structure. This line/sample location is then the output location for the corresponding input line/sample and the current elevation (current grid line/sample input locations).

b) Calculate Jitter Sensitivity Coefficients The forward model is invoked multiple times at each grid intersection to compute the effect that small attitude perturbations about each spacecraft axis have on the input space to output space line/sample mapping. This is done at each grid point as follows:

Save the current grid point input line/sample as in_line/in_samp and the current grid point output line/sample as line0/samp0.

For each spacecraft axis (roll-pitch-yaw) :

9. Perturb the attitude about the selected axis by 1 microradian.
10. Use the forward model to compute the output line/sample corresponding to the current input line/sample using the perturbed attitude and store the result in line[0]/samp[0].
11. Perturb the input line number by 1 line (delta_line = 1) and recompute the corresponding output line/sample, storing the result in line[1]/samp[1].
12. Restore the input line number to in_line and perturb the input sample number by 1 sample (delta_samp = 1) and recompute the corresponding output line/sample, storing the result in line[2]/samp[2].
13. Calculate the output space to input space line/sample sensitivities as:
    a. delta_oline_per_iline = (line[1]–line[0]) / delta_line
    b. delta_oline_per_isamp = (line[2]–line[0]) / delta_samp
    c. delta_osamp_per_iline = (samp[1]–samp[0]) / delta_line
    d. delta_osamp_per_isamp = (samp[2]–samp[0]) / delta_samp
14. Invert the resulting 2-by-2 sensitivity matrix to find the input line/samp per output line/samp sensitivities:
    a. determinant = delta_oline_per_iline * delta_osamp_per_isamp – delta_oline_per_isamp * delta_osamp_per_iline
    b. delta_iline_per_oline = delta_osamp_per_isamp / determinant
    c. delta_iline_per_osamp = -delta_oline_per_isamp / determinant
    d. delta_isamp_per_oline = -delta_osamp_per_iline / determinant
    e. delta_isamp_per_osamp = delta_oline_per_iline / determinant
15. Apply the input line/samp per output line/samp sensitivities to the output line/samp offset due to the attitude perturbation, to find the equivalent input space offset :
    a. d_iline = delta_iline_per_oline * (line[0] – line0) + delta_iline_per_osamp * (samp[0] – samp0)
    b. d_isamp = delta_isamp_per_oline * (line[0] – line0) + delta_isamp_per_osamp * (samp[0] – samp0)
16. Divide by the attitude perturbation to compute the input line/sample to attitude jitter sensitivities for this axis at this grid point:
    a. line_sens[axis] = -d_iline / perturbation
    b. samp_sens[axis] = -d_isamp / perturbation

    Where :

    line_sens[] is the array of roll-pitch-yaw line sensitivities for the grid.
    samp_sens[] is the array of roll-pitch-yaw sample sensitivities for the grid.
    perturbation is the 1 microradian attitude perturbation introduced in step 1.

    Note that the sign of the sensitivities is inverted in this calculation. This is done because the sensitivities will be used to compute the equivalent input space corrections needed to compensate for an attitude disturbance. So, since d_iline is the input space line offset that is equivalent to one microradian of jitter for the current axis, an offset of –d_iline will compensate for this jitter.

A 2-by-3 array containing the line and sample sensitivity coefficients for the roll, pitch, and yaw axes is stored for each grid point.

c) Calculate Map Coefficients

Bilinear mapping coefficients for each grid cell are calculated for mapping from input location to output location (forward mapping) and for mapping from output location to input location (inverse mapping). A separate mapping function is used for lines and samples. This equates to four mapping functions. A set of four mapping functions is calculated for each grid cell, for each SCA, for every band, and for every elevation plane that is stored in the grid.

The following methodology is used for calculating one set of four bilinear mapping equations:

A 9x4 matrix is used to fit nine points within a grid cell. The matrix equation takes the form of:

$$[A][coeff] = [b]$$

In this equation, matrix A is 9x4, vector b is 9x1, and the coefficient matrix is 4x1. The coefficient matrix, [*coeff*], can be solved to obtain the mapping coefficients as:

$$[coeff] = \left[A^T A\right]^{-1}\left[A^T b\right]$$

In the case of solving for an equation to map an input line and sample location to an output sample location, belonging to one grid cell, the matrices can be defined as:

$A_{n,0}$ = 1 where n=0,8
$A_{0,1}$ = upper left input sample location for current grid cell
$A_{1,1}$ = upper right input sample location for current grid cell
$A_{2,1}$ = lower left input sample location for current grid cell
$A_{3,1}$ = lower right input sample location for current grid cell
$A_{4,1} = (A_{0,1}+A_{1,1}+A_{2,1}+A_{3,1})/4$
$A_{5,1} = (A_{0,1}+A_{1,1})/2$
$A_{6,1} = (A_{1,1}+A_{3,1})/2$
$A_{7,1} = (A_{2,1}+A_{3,1})/2$
$A_{8,1} = (A_{2,1}+A_{0,1})/2$
$A_{0,2}$ = upper left input line location for current grid cell
$A_{1,2}$ = upper right input line location for current grid cell
$A_{2,2}$ = lower left input line location for current grid cell
$A_{3,2}$ = lower right input line location for current grid cell
$A_{4,2} = (A_{0,2}+A_{1,2}+A_{2,2}+A_{3,2})/4$
$A_{5,2} = (A_{0,2}+A_{1,2})/2$
$A_{6,2} = (A_{1,2}+A_{3,2})/2$
$A_{7,2} = (A_{2,2}+A_{3,2})/2$
$A_{8,2} = (A_{2,2}+A_{0,2})/2$
$A_{n,3} = A_{n,1}*A_{n,2}$ where n=0…8

b_0 = upper left output sample location for current grid cell
b_1 = upper right output sample location for current grid cell
b_2 = lower left output sample location for current grid cell
b_3 = lower right output sample location for current grid cell
$b_4 = (b_0+b_1+b_2+b_3)/4$

$b_5 = (b_0+b_1)/2$
$b_6 = (b_1+b_3)/2$
$b_7 = (b_2+b_3)/2$
$b_8 = (b_2+b_0)/2$

The line and sample locations listed above are defined at the grid cell corners coordinates. The points interpolated in between the grid cell line segments provide stability for what could be, most notably a mapping that involves a 45^o rotation, an ill-defined solution if only four points were used in the calculation. The set of coefficients define a bilinear mapping equation of the form:

$$sample_o = coeff_0 + coeff_1 * sample_i + coeff_2 * line_i + coeff_3 * sample_i * line_i$$

where:
$sample_o$ = output sample location
$sample_i$ = input sample location
$line_i$ = input line location

The forward mapping equations, mapping input line and sample locations to output line locations can be solved by swapping output line locations for output sample locations in the matrix [b]. The reverse mapping equations, mapping output locations to input line and sample, can be found by using output line and sample locations in the [A] matrix and the corresponding input sample and then line locations in the [b] matrix.

c.1) Calculate Forward Mappings
This function, given grid points in both input and output space, uses the Calculate Map Coefficients algorithm described above to generate the mapping polynomial coefficients needed to convert from a line/sample in input space (satellite) to one in output space (projection). It generates these coefficients for every cell in the grid.

c.2) Calculate Inverse Mappings
This function, given grid points in both input and output space, uses the Calculate Map Coefficients algorithm described above to generate the mapping polynomial coefficients needed to convert from a line/sample in output space (projection) to one in input space (satellite). It generates these coefficients for every cell in the grid.

Stage 5 - Finalize the Grid
The final stage of grid processing generates the global (rough) mapping coefficients, used to initially identify the appropriate grid cell, and computes the parallax sensitivity coefficients, used to correct for even/odd detector offset effects, for each grid cell.

a) Calculate Rough Mapping Coefficients
This routine will find the rough mapping coefficients for the grid. The rough polynomial is a set of polynomials used to map output line and sample locations to input line and sample locations. The rough polynomial is generated using a large number of points distributed over the entire scene, and by calculating a polynomial equation that maps an output location to an input location. The rough polynomial is only meant to get a “close” approximation to the input line and sample location for a corresponding output line and sample location. Once this approximation is made, the value can be refined to get a more accurate solution. A rough mapping polynomial is found for every SCA, for every band, and for every elevation plane that is stored in the grid file.

Bilinear mapping was found to be sufficient for the rough mapping. The mapping function therefore looks like the ones used for each individual grid cell. However, the set up of the matrices to solve for the mapping coefficients is different:

$$\underset{Nx4}{[A]}\underset{4x1}{[coeff]} = \underset{Nx1}{[b]}$$

Where the matrix [A] is defined by the output line and sample locations, matrix [b] is defined by either the input lines or input samples, and N is equal to the total number of points stored in the grid for one elevation plane, of one band, for a single SCA. The rough polynomial is therefore found by using all the point locations stored in the grid for a given band and elevation plane for a single SCA. There is one mapping for output line and sample location to input sample location and one mapping for output line and sample location to input line location.

Grid Cell Polynomial

This utility function calculates a "rough" mapping of output to input lines/samples. The coefficients returned from this function are used as a first order approximation to an inverse line-of-sight model. This polynomial is used to initially locate the grid cell to be used in the resampling process, providing a starting point for the more accurate inverse model based on individual grid cell parameters.

b) Calculate Detector Offsets

This function computes the detector offset values and stores linear mapping coefficients associated with detector offsets in the grid structure. Using the zero elevation plane, for each band and each SCA, loop on the input lines and samples calculating the detector offsets. The detector offsets are set up to account for the geometric differences between the primary and redundant rows of detectors and the “nominal” set of detectors modeled by the Legendre polynomials (see the TIRS LOS Model Creation ADD). These differences are considered to be consistent between actual and nominal detectors when they occur under the same acquisition conditions, i.e. they are slowly varying. These actual to nominal detector differences are due to the imperfect trade-off between space (detector offset) and time (detector delay) that is made when we temporally shift (through the use of Level 1R image fill) the deselected/replaced detectors to compensate for their spatial offsets on the focal plane. The degree to which this time/space trade is imperfect varies with height and, so, the corrections derived here and stored in the grid structure, are functions of detector offset and height.

There are also the sub-pixel detector specific offsets that are stored in the CPF. These "exact" detector specific offsets are accounted for in the resampling process. Note that the potential for deselected detectors has made it necessary to also store per-detector full-pixel offsets in the CPF (and LOS model). As a result, this detector offset sensitivity logic computes the offset sensitivity per pixel of detector offset rather than a fixed value. The routine ols2ils listed below, used for mapping an output line and sample to an input line and sample using the geometric grid, is discussed in the TIRS Image Resampling ADD.

Loop on number of bands stored in grid
    Loop on number of SCAs stored in grid
        Loop on lines and samples stored in the grid

            Get the maximum detector offset value for this band from the CPF.

Calculate the output line/sample location for the current input line and sample and the zero elevation plane, calculated using the forward model (see below) with the detector location set to MAXIMUM. This new detector type is the same as ACTUAL but uses the maximum detector offset rather than the detector-specific value.

Map calculated output line/sample back to input space using the TIRS LOS grid and ols2ils.

Delta line/sample per pixel of offset are calculated by:

$\Delta line_0$ = (nominal line - mapped line) / max offset
$\Delta sample_0$ = (nominal sample - mapped sample) / max offset

where:
nominal line = current grid cell line location
mapped line = input line location from ols2ils mapped "maximum" output line
nominal sample = current grid cell sample location
mapped sample = input sample location from ols2ils mapped "maximum" output sample
max offset = detector offset used in the MAXIMUM forward model calculations

These delta lines and samples represent the input space correction necessary to compensate for the difference between nominal and actual detectors per pixel of detector offset, for the zero elevation plane.

Repeat these calculations for the maximum elevation plane to compute $\Delta line_H$ and $\Delta sample_H$ where H is the elevation corresponding to the maximum z-plane.

Compute the line and sample even/odd offset sensitivity coefficients:

$c_0 = \Delta line_0$
$c_1 = (\Delta line_H - \Delta line_0) / H$
$d_0 = \Delta sample_0$
$d_1 = (\Delta sample_H - \Delta sample_0) / H$

Note that c_0 and d_0 are in units of pixels per pixel and c_1 and d_1 are in units of pixels per meter per pixel.

These c_i and d_i coefficients are stored in the projection grid to be used during the resampling process.

Output Line/Sample to Input Line/Sample

This utility routine maps an output space line/sample back into its corresponding input space line/sample. This is done using the "rough" polynomial from the grid to determine an initial guess at an input space line and sample. From this initial guess a grid cell row and column is calculated and

the inverse coefficients for that cell are retrieved from the grid. These coefficients are used to determine an exact input space line and sample (in extended space).

Find Grid Cell
This utility function finds the correct cell that contains the output line/sample. It finds the correct grid cell containing the output pixel by first determining the set of grid cells to be checked. It then calls a routine to perform a "point in polygon" test on each of these grid cells to determine if the pixel does indeed fall within that grid cell.

Forward Model
Having described the grid generation procedure we now turn to the forward model, referred to extensively above, in more detail.

For a given line, sample and band, propagate the forward model to determine a latitude and longitude for the specified point. This involves finding the time of the observation, constructing the instrument line-of-sight, calculating the spacecraft attitude and ephemeris for the observation time, and intersecting the projected line-of-sight with the Earth's surface. The entire forward model procedure is referred to as LOS projection and is described step by step below.

a) Project TIRS LOS
This function finds the position where the line of sight vector intersects the Earth's surface. It invokes the following sub-algorithms:

a.1) Find Time
This function finds the time into the scene given the line, sample, and band. The input sample number is 0-relative and relative to the SCA. The accounting for the primary/redundant detector offsets is based on the value of the dettype variable which may be NOMINAL, ACTUAL, MAXIMUM or EXACT. Note that the EXACT selection is treated the same as ACTUAL. This is due to the fact that even though fractional-pixel detector offsets can occur, the compensating time shifts implemented by inserting fill pixels can only be introduced in whole-line increments. So, the sub-pixel difference between the ACTUAL and EXACT detector types affects only the LOS angle not the time. The MAXIMUM detector type represents a theoretical offset that is used to calculate the parallax coefficients within the grid. This maximum is stored as #define in the prototype code, called MAX_DET_DELAY.

Due to the multiple detector rows and the potential for bad detector replacement, a nominal and an actual time can be found in a scene. If the current position within the image is given as a line and sample location, the two different "types" of times for TIRS pixels are calculated by:

```
if detector type is set to MAXIMUM
    detector_shift_x = maximum_detector_shift
    l0r_fill_pixels = round(detector_shift_x) + nominal_fill
else
    detector_shift_x = shift stored in geometric model
     l0r_fill_pixels = Fill from L0rp (also stored in geometric model)

time_index = line_number - l0r_fill_pixels
if ( time_index < 0 ) time_index = 0
if (time_index > (num_time_stamps - 1)) time_index = num_time_stamps - 1
```

actual_time = line_time_stamp[time_index] - integration_time/2
+ (line_number - l0r_fill_pixels - time_index) * TIRS_sample_time

nominal_time = actual_time + (l0r_fill_pixels – nominal_fill) * TIRS_sample_time

where:

- line_number is the zero-referenced TIRS image line number (N).
- nominal_fill is the amount of detector alignment fill to be inserted at the beginning of pixel columns that correspond to nominal detectors; that is, those detectors with a delay value of zero that are the basis for the Legendre polynomial LOS model. This value comes from the CPF and will be zero if there are no bad detectors to replace.
- l0r_fill_pixels is the total amount of detector alignment fill to be inserted at the beginning of the pixel column associated with the current detector. It includes both the nominal_fill_pixels and the detector-specific delay fill required to align deselected/replaced detectors.
- num_time_stamps is the total number of time codes (image lines) in the image. It is tested to ensure that time_index, the line_time_stamp index, does not go out of bounds.
- detector_shift_x is the amount of detector offset for the current detector from the TIRS LOS model detector delay table. It is rounded to the nearest integer pixel because time offsets can only occur in whole line increments.

The detector_shift_x offset parameter from the LOS model detector delay table is rounded to include the effects of detector deselect/replacement but not the detector-specific sub-pixel offsets.

a.2) Find TIRS LOS

This function finds the line of sight vector in sensor coordinates, using the Legendre polynomial LOS model and the SSM model stored in the TIRS LOS model, as follows:

Find normalized detector for Legendre polynomial:

$$\text{normalized detector} = \frac{2*(\text{current detector})}{(\text{number of detectors} - 1)} - 1$$

where:

current detector = sample location (in the range 0 to number of detectors-1)
number of detectors = number of detectors (samples) for current band and SCA (from TIRS LOS model)

Find across track (y) and along track (x) angles:

$$x = coef\_x_0 + coef\_x_1 * (\text{nd}) + coef\_x_2 * (1.5 * (\text{nd})^2 - 0.5) + coef\_x_3 * (\text{nd}) * (2.5 * (\text{nd})^2 - 1.5)$$

$$y = coef\_y_0 + coef\_y_1 * (\text{nd}) + coef\_y_2 * (1.5 * (\text{nd})^2 - 0.5) + coef\_y_3 * (\text{nd}) * (2.5 * (\text{nd})^2 - 1.5)$$

where:

nd = normalized detector
coef_x = Legendre coefficients for along track direction
coef_y = Legendre coefficients for across track direction

(Note: coef_x and coef_y are read from the CPF and stored in the LOS model)

If LOS requested is ACTUAL, add the whole pixel detector shift (detector, band, and SCA dependent) from the TIRS LOS model. This detector shift is only in the along track direction. Note that the TIRS LOS model contains the combined whole pixel and sub-pixel detector offset, so it must be rounded to the integer part for the ACTUAL detector type and left unrounded for the EXACT detector type.

x = x + round(detector_shift_x) * IFOV

If LOS requested is EXACT, then add individual detector offsets (detector number, band, and SCA dependent). This detector shift is in both the along and across track directions. These values are stored within the TIRS LOS model.

x = x + (detector_shift_x) * IFOV
y = y + (detector_shift_y) * IFOV

Note that the detector_shift_y parameter, from the TIRS LOS model detector delay table, is always sub-pixel. See TIRS LOS Model Creation ADD for further explanation of NOMINAL/ACTUAL/EXACT line of sight.

If the LOS request is MAXIMUM then add the maximum detector offset.

x = x + (maximum_detector_shift_x) * IFOV

Calculate LOS vector.

$$[\mathrm{los}] = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Normalize LOS.

$$\vec{\mathrm{los}} = \frac{\vec{\mathrm{los}}}{\left\|\vec{\mathrm{los}}\right\|}$$

Apply the TIRS telescope alignment matrix (from the TIRS LOS model) to the LOS vector.

$$\mathbf{[los_{SSM}]} = \mathbf{M_{Tele2SSM}}\ \mathbf{[los]}$$

Calculate the SSM angle (from the TIRS LOS model) using the sample time returned from the find time sub-algorithm (nominal or actual). Note that both the image and SSM times are referenced to the same UTC epoch (see TIRS LOS Model Creation ADD for details).

1. In the TIRS SSM model, find the last SSM angle sample, θ_n, with a time, t_n, earlier than the pixel time.
2. Linearly interpolate the SSM angle, θ, at the image time, t, from θ_n, t_n, θ_{n+1}, and t_{n+1}.
   $\theta = \theta_n * (t_{n+1} - t)/(t_{n+1} - t_n) + \theta_{n+1} * (t - t_n)/ (t_{n+1} - t_n)$
3. Compute the SSM reflection matrix.

$$\mathbf{M}_{SSM}(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta\cos\theta & \cos^2\theta & \sin\theta \\ \sin^2\theta & -\sin\theta\cos\theta & \cos\theta \end{bmatrix}$$

4. Apply the SSM reflection matrix to the LOS.
   [los$_{TIRS}$] = M$_{SSM}$(θ) [los$_{SSM}$]

a.3) Find Attitude
This function finds the precise roll, pitch and yaw at the specified time. This routine uses the "corrected" version of the attitude data stored in the TIRS LOS model. This attitude data sequence includes the effects of ground control point precision correction (if any).

Find the current time relative to attitude data start time stored in the LOS model.

dtime = time + image epoch time – attitude epoch time

Note:
time = nominal time of input sample relative to the start of the image epoch time = image start time from LOS model, only need seconds of day field since all epochs are adjusted to the same day.
attitude epoch time = attitude data start time from LOS model, only need seconds of day field since all epochs are adjusted to the same day.

Find index into attitude data (stored in model) corresponding to dtime:

$$\text{index} = \text{floor}\left(\frac{\text{dtime}}{\text{attitude sampling rate}}\right)$$

where:
attitude sampling rate = sample period from LOS model

This attitude index determination could also be implemented as a search through the attitude data time stamps which are stored in the LOS model. The selected index would be the index of the last time that does not exceed dtime.

Attitude is found by linearly interpolating between the attitude values located at index and index+1 using the corrected attitude sequence from the LOS model:

$$w = \frac{\text{fmod}(\text{dtime}, \text{attitude sampling rate})}{\text{attitude sampling rate}}$$

$$\text{roll} = \text{model roll}_{index} + (\text{model roll}_{index+1} - \text{model roll}_{index}) * w$$
$$\text{pitch} = \text{model pitch}_{index} + (\text{model pitch}_{index+1} - \text{model pitch}_{index}) * w$$
$$\text{yaw} = \text{model yaw}_{index} + (\text{model yaw}_{index+1} - \text{model yaw}_{index}) * w$$

a).3.i. Find Jitter Find the high frequency roll, pitch and yaw corrections at the specified input image line/sample coordinate. This routine uses the jitter table stored in the TIRS LOS model. This table is

time aligned with the TIRS image line sampling times, so the jitter table look-up proceeds directly from the input line/sample coordinates:

Find the current detector number from the input sample location:
detector = round(sample)
Verify that the detector is in the valid range for this band (return error if not).
Look up the number of L0R fill pixels for this detector (from the fill table).
Calculate the jitter table index:
Index = round(line) – l0r_fill_pixels
Verify that jitter table index is within the valid range for the table (return zeros if not).
Extract the roll-pitch-yaw jitter values for the current index from the jitter table and return these values.

Note that the jitter values are a direct look-up without interpolation. This does not compromise accuracy because this function is only used for cases of EXACT detector projection (e.g., the TIRS data simulator) for which the input line/sample coordinates are integers. The jitter values extracted by Find Jitter are added to the low frequency roll-pitch-yaw values interpolated by Find Attitude, by the calling procedure, Project TIRS LOS, when the EXACT option is in force.

a.4) Move Satellite Sub-Algorithm

This function computes the satellite position and velocity at a delta time from the ephemeris reference time using Lagrange interpolation. This is a utility sub-algorithm that accesses the "corrected" version of the model ephemeris data to provide the TIRS position and velocity at any specified time. Since the model ephemeris arrays are inputs to this sub-algorithm it will work with either the ECI or ECEF ephemeris data.

Calculate time of current line/sample relative to start time of ephemeris start time.

reference time = time + image epoch time – ephemeris epoch time

where:
time = nominal time of input sample relative to the start of the imagery
image epoch time = image start time from LOS model, only need seconds of day since all epochs are on same day.
ephemeris epoch time = ephemeris start time from LOS model, only need seconds of day since all epochs are on same day.

Find index into ephemeris data stored in model.

$$\text{index} = \text{floor}\left(\frac{\text{reference time}}{\text{ephemeris time steps}} - \frac{\text{number of Lagrange points}}{2} \right)$$

where:
ephemeris time steps = time between ephemeris samples
number of Lagrange points = number of points to use in Lagrange interpolation

Use Lagrange interpolation to calculate satellite position and velocity in ECEF (or ECI, depending on which sequence is provided) coordinates at time of current line/sample.

X = Lagrange(model satellite ECEF/ECI x[index])

Y = Lagrange(model satellite ECEF/ECI y[index])
Z = Lagrange(model satellite ECEF/ECI z[index])
XV = Lagrange(model satellite ECEF/ECI vx[index])
YV = Lagrange(model satellite ECEF/ECI vx[index])
ZV = Lagrange(model satellite ECEF/ECI vx[index])

where:
X = satellite x coordinate
Y = satellite y coordinate
Z = satellite z coordinate
XV = satellite x velocity
YV = satellite y velocity
ZV = satellite z velocity

a.5) Convert Sensor LOS to Geocentric

This function finds the line of sight vector from the spacecraft to a point on the ground by transforming the line of sight vector in sensor coordinates to perturbed spacecraft coordinates.

Use the TIRS alignment matrix in the TIRS LOS model to convert the TIRS LOS vector from sensor to body. Then apply roll, pitch, and yaw to the LOS to convert body to orbital. Finally, use the ephemeris to construct the orbital to ECEF rotation matrix and use it to transform LOS to ECEF.

First, using the 3x3 ACS to instrument alignment transformation matrix stored in the TIRS LOS model, calculate the instrument to ACS transformation matrix.

$$[\text{Instrument to ACS}] = [\text{ACS to Instrument}]^{-1}$$

Transform LOS from Instrument to ACS/body coordinates.

$$[\text{navigation los}] = [\text{Instrument to ACS}][\text{los}]$$

Calculate attitude perturbation matrix using interpolated attitude values. Note that these values include the effects of precision LOS correction (if any) as these will be built into the "corrected" attitude stream in the LOS model. Also note that for Earth-view acquisitions the roll-pitch-yaw values will be with respect to the orbital coordinate system but for celestial acquisitions they will be with respect to ECI.

Calculate perturbation matrix, [perturbation], due to roll, pitch, and yaw:

$$[\text{attitude perturbation}] = [Y_{yaw}][P_{pitch}][R_{roll}] =$$

$$\begin{bmatrix} \cos(p)\cos(y) & \sin(r)\sin(p)\cos(y)+\cos(r)\sin(y) & \sin(r)\sin(y)-\cos(r)\sin(p)\cos(y) \\ -\cos(p)\sin(y) & \cos(r)\cos(y)-\sin(r)\sin(p)\sin(y) & \cos(r)\sin(p)\sin(y)+\sin(r)\cos(y) \\ \sin(p) & -\sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}$$

Calculate new LOS in orbital coordinates (Earth-view) or ECI (celestial) due to attitude perturbation:

$$[\text{perturbation los}] = [\text{perturbation}][\text{navigation los}]$$

For Earth-view acquisitions, calculate the transformation from Orbital Coordinates to ECEF. The position and velocity vectors used in calculating the transformation are those calculated above. These vectors are in ECEF allowing the LOS to be transformed from the instrument coordinate system to the ECEF coordinate system.

Transform perturbed LOS from Orbital to ECEF.

$$[\text{ECEF los}] = [\text{ORB2ECEF}][\text{perturbation los}]$$

For celestial acquisitions, the ECI los ([perturbation los]) is returned.

a.6) Find Target Position

This function finds the position where the line of sight vector intersects the Earth's surface.

Intersect the LOS in ECEF with the Earth model calculating the target ECEF vector. The ECEF vector is then used to compute the geodetic latitude and the longitude of the intersection point.

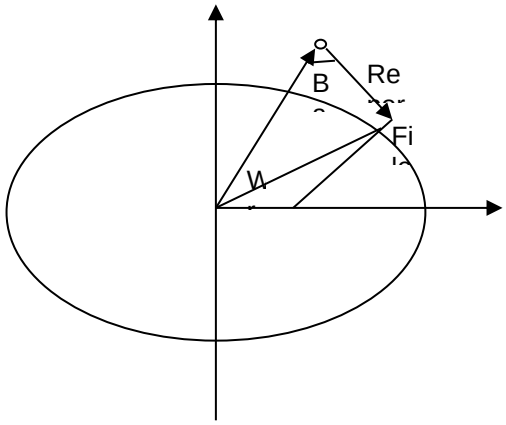


Figure 7: Intersecting LOS with Earth model

Where:

rs = satellite position vector
re = geocentric Earth vector
los = line-of-sight vector

Intersect LOS with ellipsoid

f) Rescale vectors with ellipsoid parameters.

$$rs' = \left[\frac{rsx}{a} \quad \frac{rsy}{a} \quad \frac{rsz}{b} \right]$$

$$re' = \left[\frac{rex}{a} \quad \frac{rey}{a} \quad \frac{rez}{b} \right]$$

$$los' = \left[\frac{losx}{a} \quad \frac{losy}{a} \quad \frac{losz}{b} \right]$$

where:

a = semi-major axis of Earth ellipsoid
b = semi-minor axis of Earth ellipsoid
rs' = rescaled satellite position vector
re' = rescaled geocentric Earth vector
los' = rescaled LOS vector

g) From the Law of Cosines

$$|re'|^2 = |d * los'|^2 + |rs'|^2 - 2|d * los'||rs'|\cos(\delta)$$

where:

d = los' vector length
δ = angle between rs' and los'

$$\cos(\delta) = \frac{los' \bullet rs'}{|los'||rs'|}$$

By definition | re' | = 1

Rearranging the equation determined from the Law of Cosines in terms of the constant d.

$$d^2|los'|^2 + 2d(los' \bullet rs') + |rs'|^2 - 1$$

Solving for d using the quadratic equation.

$$d = \frac{-|los' \bullet rs'| - \sqrt{|los' \bullet rs'|^2 - |los'|^2\left(|rs'|^2 - 1\right)}}{|rs'|^2}$$

h) Compute new target vector.

$$re' = rs' + d * los'$$

i) Rescale target vector.

$$re = \left[a * rex' \quad a * rey' \quad b * rez' \right]$$

j) Compute Geodetic coordinates (see Geocentric to Geodetic below).

$$(rex, rey, rez) \Rightarrow (\varphi_0, \lambda_0, h_0)$$

If target height (H), or elevation corresponding to current z plane, is not zero:

Initialize:
Target vector: rt=re
Target height: h_0=0

Iterate until Δh =(h_i-H) is less than TOL

g) Calculate delta height.

$$\Delta h = h_i - H$$

h) Compute length of LOS.

$$d = \sqrt{(rtx - rsx)^2 + (rty - rsy)^2 + (rtz - rsz)^2}$$

where:
d = length of LOS vector
rt = target vector
rs = spacecraft position vector

i) Compute LOS /height sensitivity.

$$q = n \bullet los$$

Where **n** is a vector normal to the ellipsoid surface.

$$n = \left[\cos(\varphi_i)\cos(\lambda_i) \quad \cos(\varphi_i)\sin(\lambda_i) \quad \sin(\lambda_i)\right]^T$$

and:
q = LOS height sensitivity coefficient
los = LOS unit vector
ϕ_i = current estimate of ground point latitude
λ_i = current estimate of ground point longitude

j) Adjust LOS.

$$d = d + q * \Delta h$$

k) Re-compute target vector.

$$rt = rs + d * los$$

l) Calculate new geodetic coordinates and corresponding height above ellipsoid.

$$(rtx, rty, rtz) \Rightarrow (\varphi_{i+1}, \lambda_{i+1}, h_{i+1})$$

Calculate the geodetic latitude and longitude from the final ECEF vector.

a.7) Geocentric to Geodetic

The relationship between ECEF and geodetic coordinates can be expressed simply in its direct form:

$e^2 = 1 - b^2 / a^2$
$N = a / (1 - e^2 \sin^2(\varphi))^{1/2}$
$X = (N + h) \cos(\varphi) \cos(\lambda)$
$Y = (N + h) \cos(\varphi) \sin(\lambda)$
$Z = (N (1-e^2) + h) \sin(\varphi)$

where:

X, Y, Z	-	ECEF coordinates
φ, λ, h	-	Geodetic coordinates (lat ϕ, long λ, height h)
N	-	Ellipsoid radius of curvature in the prime vertical
e^2	-	Ellipsoid eccentricity squared
a, b	-	Ellipsoid semi-major and semi-minor axes

The closed-form solution for the general inverse problem (which is the problem here) involves the solution of a quadratic equation and is not typically used in practice. Instead, an iterative solution is used for latitude and height for points that do not lie on the ellipsoid surface, i.e., for h ≠ 0.

To convert ECEF Cartesian coordinates to spherical coordinates:

Define:

$$\text{radius} = \sqrt{X^2 + Y^2 + Z^2}$$

$$\varphi' = \sin^{-1}\left(\frac{Z}{\text{radius}}\right)$$

$$\lambda = \tan^{-1}\left(\frac{Y}{X}\right)$$

Initialize:

$$\theta = \varphi'$$

$$h_0 = 0$$

Iterate until $abs(h_i - h_{i+1}) < TOL$

$$\mathrm{re} = \frac{a * \sqrt{1-e}}{\sqrt{1-e*\cos^2(\theta)}}$$

$$\varphi = \tan^{-1}\left(\frac{\tan(\theta)}{1-e}\right)$$

$$\Delta\varphi = \varphi - \theta$$

$$\mathrm{rs} = \mathrm{radius}^2 - \mathrm{re}^2 * \sin^2(\Delta\varphi)$$

$$\mathrm{h}_{i+1} = \sqrt{\mathrm{rs}} - \mathrm{re} * \cos(\Delta\varphi)$$

$$\theta = \varphi' - \sin^{-1}\left(\frac{\mathrm{h}_{i+1}}{\mathrm{radius} * \sin(\Delta\varphi)}\right)$$

Projection Transformation
This function converts coordinates from one map projection to another. The transformation from geodetic coordinates to the output map projection depends on the type of projection selected. The mathematics for the forward and inverse transformations for the Universal Transverse Mercator (UTM), Lambert Conformal Conic, Transverse Mercator, Oblique Mercator, Polyconic, and the Space Oblique Mercator (SOM) map projections are handled by U.S Geological Survey's (USGS) General Cartographic Transformation Package (GCTP), as noted below.

Projection Errors
This function reports projection transformation package errors. The function receives a GCTP error code and prints the correct error message.

General Cartographic Transformation Package (gctp)
Map projections are handled by U.S Geological Survey's (USGS) General Cartographic Transformation Package (GCTP), which may be obtained at http://edcftp.cr.usgs.gov/pub/software/gctpc/.

Grid Structure Summary
Tables 1 and 2 below show the detailed contents of the TIRS LOS grid structure.

Geometric Grid Structure Contents
Satellite Number (8)
WRS Path
WRS Row (may be fractional)
Acquisition Type (Earth, Lunar, Stellar)
Scene Framing Information:
Frame Type: PROJBOX, MINBOX, MAXBOX, PATH, PATH_MINBOX, PATH_MAXBOX
Projection Units (text): METERS, RADIANS, ARCSECONDS
Projection Code: GCTP integer code for UTM, SOM, etc...
Datum: WGS84
Spheroid: GCTP integer code = 12 (WGS84/GRS80)
UTM Zone: UTM zone number (or 0 if not UTM)
Map Projection Parameters: 15-element double array containing parameters

Corners: 4 by 2 array of projection coordinates for UL, LL, UR, and LR corners
Path Oriented Framing Information:
Center Point: latitude and longitude of WRS scene center
Projection Center: Map x/y of WRS scene center
Rotation Angle: Rotation (from true north) of the path frame (degrees)
Orientation Angle: Rotation (from grid north) of the path frame (degrees)
Active Image Areas: latitude and longitude (in degrees) of the four corners of the active image area (excluding leading and trailing SCA imagery) for each band
Grid Structure Information:
Number of SCAs
Number of Bands
Band List: array of band IDs included in grid
Array of band grid structures, one for each SCA in each band (see Table 2)

Table 1: TIRS LOS Grid Structure Contents

Grid Structure Contents for Each SCA in Each Band
Band number
Grid cell size: number of image lines and samples in each grid cell
Grid cell scale: 1/lines per cell and 1/samples per cell
Pixel size: in projection units (usually meters)
Number of lines in output image
Number of samples in output image
Number of lines in grid (NL)
Number of samples in grid (NS)
Number of z-planes (NZ)
Index of zero-elevation z-plane
Z-plane spacing: elevation increment between z-planes
1D array of input line numbers corresponding to each grid row
1D array of input sample numbers corresponding to each grid column
3D array of output lines for each grid point (row-major order) (NS*NL*NZ)
3D array of output samples for each grid point (row-major order) (NS*NL*NZ)
Array of line c_0, c_1 even/odd offset coefficients (row-major order) (2*NS*NL)
Array of sample d_0, d_1 even/odd offset coefficients (row-major order) (2*NS*NL)
3D array of forward mapping (ils2ols) coefficient sets (NS*NL*NZ)
3D array of inverse mapping (ols2ils) coefficient sets (NS*NL*NZ)
3D array of line jitter sensitivity coefficient vectors (3*NS*NL*NZ)
3D array of sample jitter sensitivity coefficient vectors (3*NS*NL*NZ)
Degree of rough polynomial
Array of rough line polynomial coefficients ((degree+1)^2 * NZ values)
Array of rough sample polynomial coefficients ((degree+1)^2 * NZ values)

Table 2: Per Band LOS Grid Structure Contents

LOS Projection Grid Size

To fully capture the potential variability of the 50 Hz attitude data would require a grid spacing of 1.4 TIRS lines. This is impractical. The TIRS error budgets assumed that attitude variations at frequencies up to 10 Hz would be corrected in the LOS model. Such variations can be captured by sampling at 20 Hz or higher. This corresponds to a grid spacing of 3.5 lines, which is still not terribly practical. A nominal grid spacing of 5 lines was initially adopted for TIRS. Grid size is not the concern it is for OLI data as the TIRS images are substantially smaller (only 2 bands, 3 SCAs and ~2071 lines) but the practicalities of working with such a dense grid (e.g., on the grid cell search logic) make it desirable to implement per-line high frequency correction logic as this permits the use of a sparser grid. The inclusion of a high frequency jitter table in the TIRS model and jitter sensitivity coefficients in the grid structure allow the grid to be less dense in the time (line) dimension. The baseline assumption is that attitude frequencies above 1 Hz will be relegated to the jitter table allowing the TIRS grid density to be reduced to 10 lines thus saving grid space even with the addition of the new jitter sensitivity fields.

7.3.2.8 Maturity

Though much of the OLI LOS projection algorithm was reused there are several areas where changes were necessary:

1. The Legendre polynomial model is higher order (3rd order instead of 2nd order).
2. The scene select mirror adds some additional logic to the Find LOS sub-algorithm.
3. The computation of image times is essentially the same as for OLI.
4. The required grid sampling rate will be higher (in terms of image lines) due to the lower TIRS sampling rate. This has made it advantageous to implement, as of version 3.1 of this algorithm, a per-line high frequency correction capability to capture attitude dynamics above 1 Hz in order to allow for a sparser LOS projection grid.
5. As with OLI, the highly accurate ephemeris provided by the spacecraft makes it worthwhile to include compensation for the offset between the spacecraft center of mass and the TIRS instrument in the TIRS LOS model.
6. As with OLI, the inertial to Earth fixed coordinate transformation logic should include leap seconds and light travel time effects.
7. The TIRS LOS projection logic includes velocity aberration and light travel time effects.

7.3.2.9 Notes

Some additional background assumptions and notes include:

1. The NOVAS planetary ephemeris file provides the lunar ephemerides used to define the reference output space for lunar image processing. This file is in the original JPL format and is provided to the NOVAS routines as an input.
2. The number of elevation planes in the grid is computed from the elevation range provided by the DEM and the maximum elevation plane spacing stored in the CPF..
3. The default grid density is hard coded (through #define statements) but these values are overwritten by values read from the CPF.
4. The "thresholds and limits" parameters, stored either in system tables or the database for L7 and ALI, have been moved to the CPF. This makes date specific changes, e.g., due to a change in the nominal orbit during early- or late-mission operations, easier to manage.
5. The problem of multiple terrain intersections needs to be addressed for off-nadir images, though probably not for purposes of grid generation, since it requires analysis of the full resolution DEM. This problem is being handled for OLI images by a new processing step which creates an output space mask of pixels that are obscured by terrain. This is not as much of a

concern for the (lower resolution) TIRS images and is ignored. Refer to the Terrain Occlusion ADD for details.

6. A number of data elements that are shown as coming from the TIRS LOS Model (e.g., TIRS to ACS reference alignment matrix/quaternion, spacecraft CM to TIRS offset in ACS reference frame, focal plane model parameters) are also contained in and were loaded into the model from the CPF. This is a departure from the ALIAS prototype which accesses fields of this type from the CPF wherever needed rather than merging them into the TIRS LOS model.

7.3.3 TIRS Resampling Algorithm

7.3.3.1 Background/Introduction

Since the Thermal InfraRed Sensor (TIRS) uses a pushbroom sensor architecture that is very similar to the Operational Land Imager (OLI), we can adopt a common approach to image resampling. Despite the common approach, the TIRS resampling algorithm will be maintained in a separate ADD to accommodate the substantially different OLI and TIRS development schedules. It should be possible to converge the OLI and TIRS algorithms once they achieve a similar level of maturity.

The resampling algorithm is used to take a L1R image in original sensor geometry, which has unevenly spaced pixels with respect to the surface of the object imaged, and creates a reprojected image where all image pixels are located within an evenly spaced set of grid points, or output space, with respect the object imaged. This mapping is subject to the errors associated with the interpolation method used to determine the digital numbers associated with the output image.

The TIRS geometric resampling grid and geometric model are used to calculate the mappings between the input and output space. The TIRS geometric model contains the individual detector offsets from a nominal location (i.e., departures from the Legendre polynomial line-of-sight model, offsets due to bad detector replacement) while the geometric resampling grid contains all other mapping variables. The resampling grid provides a mapping from a 2D input space to a 3D output space and vice versa. The output space corresponds to x/y/z projection locations while the input space corresponds to line/sample locations within the L1R. The z component in output space is elevation. If elevation is not to be accounted for during processing an elevation of zero is used for mapping output pixels to input pixels.

Due to what can be rather large sample-to-sample offsets within a L1R image, the cubic convolution interpolation option works in a two step process. A hybrid set of pixels in the sample direction are created using cubic convolution resampling in the line direction. This creates a set of unevenly spaced pixels in the sample direction. The Akima A interpolation method is then used to determine the final digital number for the output image by resampling the hybrid pixels in the sample direction. The nearest neighbor resampling option simply determines the closest input pixel for corresponding output pixel.

The TIRS resampling algorithm is derived from the corresponding OLI algorithm. The sensor architecture between the instruments is similar enough that a majority of the OLI algorithm can be reused. Though the TIRS focal plane geometry is somewhat simpler than the OLI due to the lack of detector stagger, it can be represented as a special case of the same model through appropriate selection of calibration parameters. The baseline geometric modeling approach will use the same 3D gridding approach for OLI and TIRS. The OLI algorithm may have to be modified to accommodate any differences in TIRS data formats.

7.3.3.2 Dependencies

The OLI/TIRS resampling algorithm assumes that the Ancillary Data Preprocessing, OLI and TIRS Line-of-Sight (LOS) Model Creation, and OLI and TIRS Line-of-Sight Projection to Ellipsoid and Terrain algorithms have been executed, and a L1R has been generated. If a digital elevation model (DEM) is given as input to account for relief, or terrain, displacement the grid must have an adequate number and range (elevation bounds) of z-planes to cover the entire elevation range within the L1R. A geometric model and grid must be available for the L1R. More information about the data structure and contents of the Geometric Model and Resampling Grid can be found in the Ancillary Data Preprocessing, OLI Line-of-Sight (LOS) Model Creation and Line-of-Sight Projection to Ellipsoid and Terrain, TIRS LOS Model Creation and LOS Projection to Ellipsoid and Terrain, Algorithm Description Documents (ADDs).

7.3.3.3 Inputs

The resampling algorithm and its component sub-algorithms use the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs
L1R Image
TIRS Resampling Grid (see the TIRS Line of Sight Projection ADD for contents)
Bands to process
Terrain correction Flag (yes/no)
DEM (if terrain flag set to yes)
SCA combine flag (yes/no)
TIRS Geometric Model (see TIRS Line of Sight Model Creation ADD for contents)
Resampling type (CC,NN)
Minimum and maximum DN of output (see note #7)
Output data type
α (if resampling type is CC) (defaults to -0.5)
Fill pixel value

7.3.3.4 Outputs

Resampled output image (L1G, L1GT or L1T)
Image data descriptor record (DDR) (See table 1)

7.3.3.5 Options

Cubic convolution or nearest neighbor resampling
Creating an output image with Sensor Chip Assemblies (SCAs) combined or separated
Applying terrain correction, yes or no

7.3.3.6 Procedure

TIRS resampling interpolates radiometrically corrected but geometrically raw image data to a map projected output space. The resampling process uses information stored in the TIRS resampling grid along with focal plane calibration data stored in the TIRS geometric model to map output projection locations to an input location. Since an input location for an output pixel typically lies at a non-integer location interpolation is used to find the pixel values associated with this non-integer location. TIRS resampling is performed on the geometrically raw L1R data using one of two methods; cubic convolution combined with the Akima A method, or nearest neighbor. Note that modulation transfer function compensation (MTFC) and bilinear resampling are not supported in the baseline algorithm. Due to the lack of inherent band registration and the need to perform sub-pixel registration to achieve TIRS band alignment, cubic convolution combined with the Akima A interpolation method will be used to generate the standard LDCM products. It is also important to have the best subpixel accuracy in the output image during geometric characterization and calibration, so cubic convolution is chosen for interpolation during the characterization and calibration of the TIRS instrument. The ALIAS-heritage nearest neighbor interpolation capability is also provided as an option for special-purpose science products and testing purposes. Since both standard product generation and geometric characterization and calibration are the focus of this document, the only interpolation method discussed in detail here is the cubic convolution combined with the Akima A method.

During resampling, there is a need to know what input pixel goes with a given output pixel. The geometric processing system does not have a “true” inverse model to perform this calculation. Instead, for a given output pixel, the corresponding input pixel is found from the forward and inverse mapping coefficients stored in the resampling grid. There are two scenarios when performing this calculation. The first involves performing resampling for a systematic image in which case the dimension for z, or elevation, is zero. This involves only a two dimensional operation in line and sample. The second involves performing resampling for a terrain corrected image. A terrain corrected image has the effects of relief removed from the output imagery. When working with a terrain corrected image, a 3-dimensional operation is performed during the inverse mapping with the dimensions being input (L1R) line, input sample, and elevation (figure 1). Both procedures of mapping output pixel locations to input pixel locations are discussed below.

Due to layout of the TIRS focal plane, there are along-track offsets between spectral bands within each SCA, along-track offsets between even and odd SCAs, and a reversal of the band ordering in adjacent SCAs. This leads to an along-track offset in the imagery coverage area for a given band between odd and even SCAs as well as an offset between bands within each SCA. To create more uniform image coverage within a geometrically corrected output product, the leading and trailing imagery associated with these offsets is trimmed. This trimming is controlled by a set of latitude/longitude bounds for the active image area for each band, contained in the input resampling grid. Trimming is implemented by converting these bounds to a look up table that lists the starting and ending sample location of active (non-fill) data for each line of the output image.

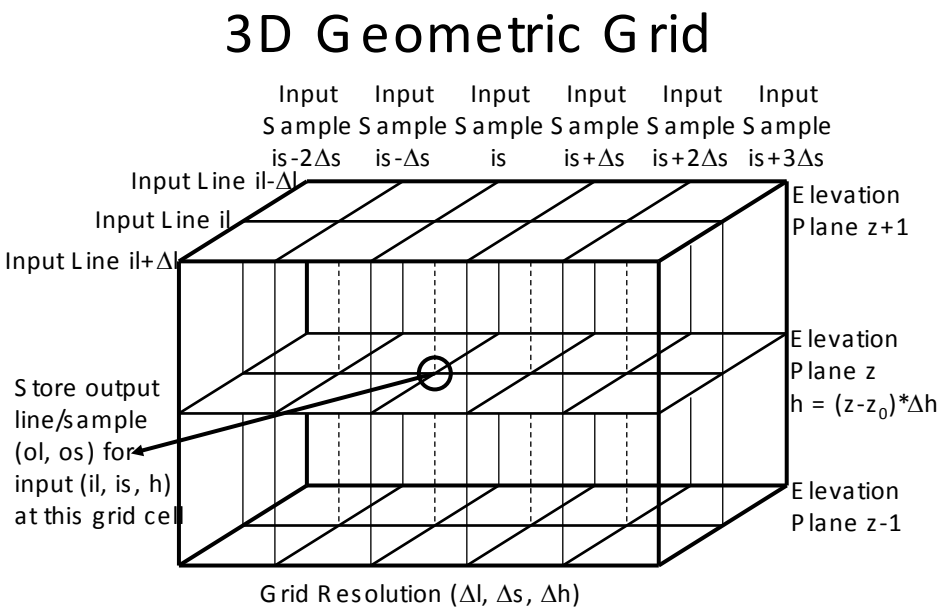


Figure 1. 3D Grid Representation

Using the geometric grid to map an output pixel location to an input pixel location.

To find an input line/sample location for an output line/sample location given that the elevation is zero:

1) Calculate an input line and sample location using the rough polynomial stored in the resampling grid and the current output line and sample location.

$$\text{approximate input line} = \sum_{n=0}^{N}\left[\sum_{m=0}^{M}\left(ra_{n,m} * (\text{outputsample})^{m}\right) * (\text{outputline})^{n}\right]$$

$$\text{approximate input sample} = \sum_{n=0}^{N}\left[\sum_{m=0}^{M}\left(rb_{n,m} * (\text{outputsample})^{m}\right) * (\text{outputline})^{n}\right]$$

Where:

ra = rough polynomial mapping coefficients for line mapping
rb = rough polynomial mapping coefficients for sample mapping
M = Number of sample coefficients in polynomial
N = Number of line coefficients in polynomial

Previous experience when working with the ALI instrument has demonstrated a 1st order polynomial in both the line and sample direction will suffice for the rough polynomial, thus M = N = 1.

$$\text{approximate input line} = a_0 + a_1 * \text{outputsample} + a_2 * \text{outputline} + a_3 * \text{outputsample} * \text{outputline}$$

$$\text{approximate input sample} = b_0 + b_1 * \text{outputsample} + a_2 * \text{outputline} + a_3 * \text{outputsample} * \text{outputline}$$

There is no evidence to believe that this will not also be the case when working with the TIRS instrument.

2) Calculate the grid cell location for the approximate input line and sample location.

$$\text{row} = \frac{\text{approximate input line}}{\text{number of lines per cell}}$$

$$\text{column} = \frac{\text{approxinate input sample}}{\text{number of samples per cell}}$$

Where:
number of lines per cell = size of grid cell in lines
number of samples per cell = size of grid cell in samples

Set this grid cell column and row location as the current grid cell column and row location.

3) Using the current grid cell location check if the correct grid cell has been found.

Use input (current) mapping grid cell coefficients (a_i and b_i) to map output line and sample to input:

input line = $b_0 + b_1 *$ output sample $+ b_2 *$ output line $+ b_3 *$ output line * output sample
input sample = $a_0 + a_1 *$ output sample $+ a_2 *$ output line $+ a_3 *$ output line * output sample

Calculate the grid cell location for this input line and sample location:

$$\text{new row} = \frac{\text{input line}}{\text{number of lines per cell}}$$

$$\text{new column} = \frac{\text{input sample}}{\text{number of samples per cell}}$$

If the new grid cell (new row and new column) is the same as the current grid cell (current row and current column):

The correct grid cell has been found, inverse grid mapping coefficients for this grid cell are used to calculate the input line/sample for the current output line/sample.

If the new grid cell (new row and new column) is not the same the current grid cell (current row and current column):

The new grid cell is chosen as current grid cell and the 3rd step is repeated until the correct grid cell is found.

This routine or function listed above, of mapping output pixel locations to input pixel locations without taking into account elevation, will be referred to as ols2ils (output space line-sample to input space line-sample mapping). The ols2ils sub-algorithm takes a given output line and sample location and calculates the grid cell column and row location along with the corresponding input line and sample location for that output location.

To find an input line/sample location for an output line/sample location given that the elevation is not zero:

1) Find the z planes that the elevation associated with the output pixel falls between.

$$z\ plane = (\text{int})\text{floor}\left(\frac{\text{elevation}}{\text{elevation increment}}\right) + z_{elev=0}$$

Where:

elevation = elevation associated with current output location (from DEM)
elevation increment = elevation increment between z planes stored in grid
$z_{elev=0}$ = zero z plane, the index of the zero elevation z-plane

The output line/sample falls between z plane and z plane+1.

2) Call ols2ils for z plane and z plane+1. This yields (input $sample_0$, input $line_0$), and (input $sample_1$, input $line_1$).

3) Interpolate between z plane and z plane + 1 to find input line and sample location for elevation.

Calculate elevations for z plane and z plane + 1:
$elev_0$ = elevation increment * (z plane - zero z plane)
$elev_1$ = $elev_0$ + elevation increment

Calculate weights for ols2ils results:

$$w_0 = \frac{elev_1 - elevation}{elev_1 - elev_0}$$

$$w_1 = \frac{elevation - elev_0}{elev_1 - elev_0}$$

input sample = input $sample_0$ * w_0 + input $sample_1$ * w_1
input line = input $line_0$ * w_0 + input $line_1$ * w_1
Where:
input $sample_0$ = input sample for z plane
input $sample_1$ = input sample for z plane + 1
input $line_0$ = input line for z plane
input $line_1$ = input line for z plane + 1

This routine or function listed above, which performs the three-dimensional output space line-sample to input space line-sample mapping, is referred to as 3d_ols2ils.

Resampling Methodology

The along and cross track detector offsets are applied during resampling. These include both the dynamic primary and redundant detector terrain-dependent relief and parallax effects that were calculated during the resampling grid generation, and the individual detector shifts that are stored in the TIRS geometric model. The nature of these geometric effects due to the individual detector characteristics is such that, in input space, they are evenly spaced in the line direction but unevenly spaced in the sample direction. This is due to the fact that as you move along raw imagery in the line direction, the detector number does not change. Since the detector number does not change along the line direction in raw input space, the along track detector offset, stored within the geometric model, does not change. These geometric effects, due to these detector offsets, are slowly varying in time staying essentially constant within the area that resampling is performed. Therefore the along track geometric effect, and essentially spacing in the line direction, can be treated as a constant over this area. The same logic helps explain why the across track detector offset is not constant in the

sample direction, since each sample comes from a different detector. This creates unevenly spaced samples in raw input space. An example of a detector layout and its' associated offset can be seen in figure 2. The squares in figure 2 represent a location of an input pixel, taking into account the detector offsets. The circle with the cross hairs in figure 2 represents the true input location for the current output pixel. It is at this point that an interpolated value is needed to represent the current output pixel.

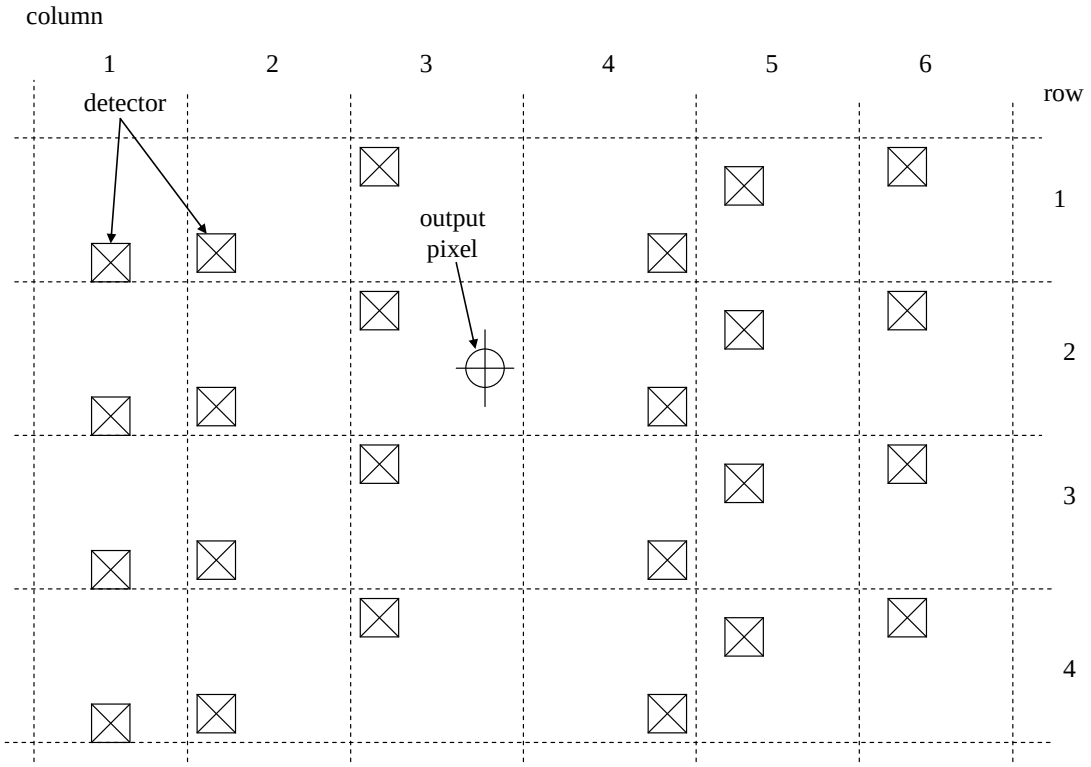


Figure 2. Example detector layout

Detector offsets are handled in the resampler by first applying a resampling kernel in the line direction that assumes evenly spaced detectors. Cubic convolution interpolation is used in the line direction; this will align a set of pixels in the sample direction. Once the pixels are aligned in the sample direction, at uneven spacing, the Akima A interpolation is used to find the final output pixel value. The linear arrangement of the TIRS detectors, not accounting for bad detector replacement, may have made it possible to avoid this complication, but treating the TIRS detectors as a special case with zero even/odd detector stagger, allows the use of a common resampling approach for OLI and TIRS.

Cubic convolution interpolation uses a set of piecewise cubic spline interpolating polynomials. The polynomials have this form:

$$f(x) = \begin{cases} (\alpha+2)|x|^3 - (\alpha+3)|x|^2 + 1 & 0 \le |x| < 1 \\ \alpha|x|^3 - 5\alpha|x|^2 + 8\alpha|x| - 4\alpha & 1 \le |x| < 2 \\ 0 & |x| > 2 \end{cases}$$

Four points, centered on the point to be interpolated, are used in interpolation. The weights for each point are generated from *f(x)*. The α in the cubic convolution function is a variable parameter that effects the edge slope of the function. For standard processing, a value of -0.5 is used. An example of what the cubic convolution function looks like, and the corresponding weights for a phase shift of zero (marked as x's), is shown in figure 3.

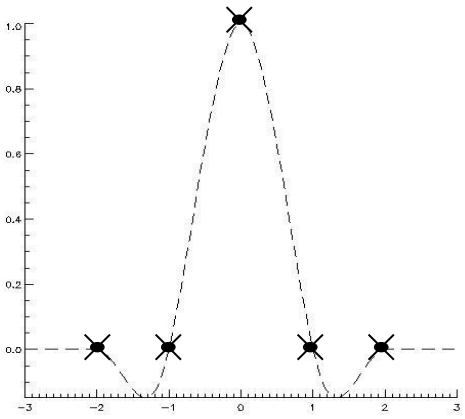


Figure 3. Cubic Convolution Function

As stated previously; for the TIRS resampler the cubic convolution resampling process produces a set of hybrid points that are aligned in the line direction. This is done by resampling several sets of L1R pixels in the line direction using the cubic convolution kernel; each time cubic convolution is performed one hybrid pixel is produced. The set of hybrid points produced from the cubic convolution process are not evenly spaced in the sample direction. Figure 4 illustrates a set of hybrid samples that have been aligned in the line direction using the cubic convolution process.

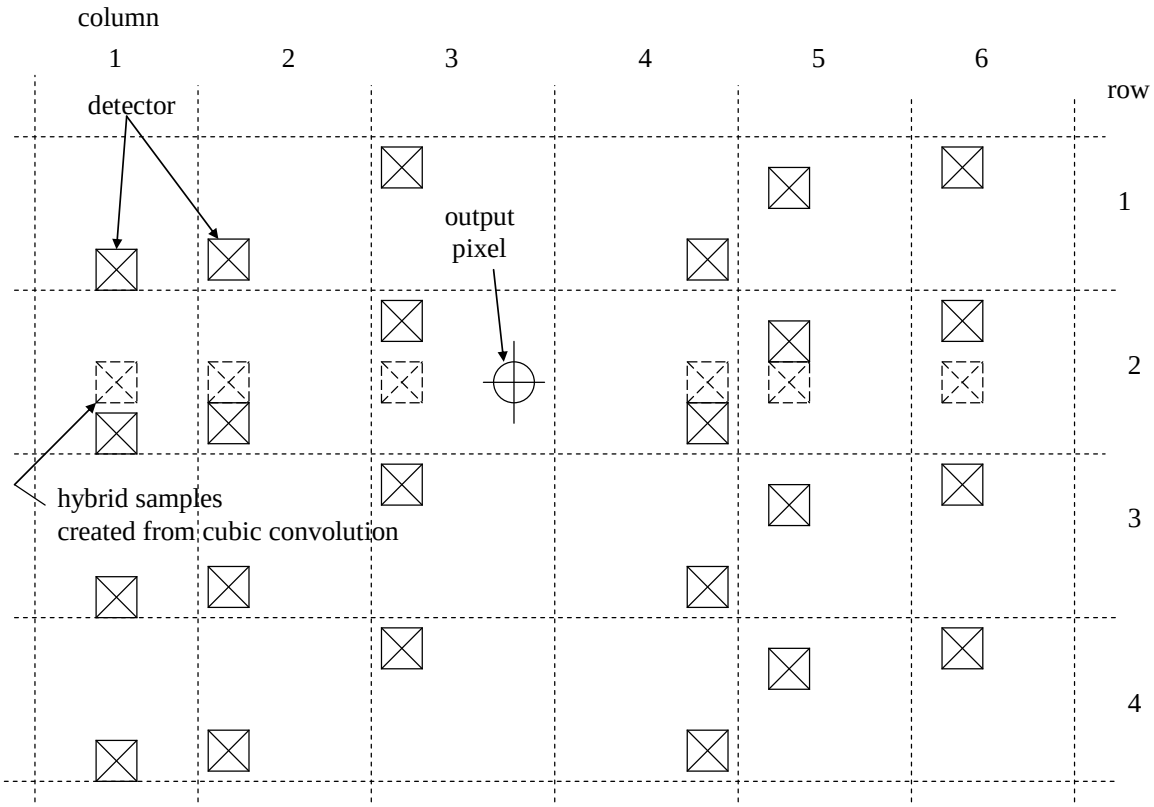


Figure 4. Hybrid pixels for detector offsets

The Akima A method for interpolation is used for interpolating the hybrid pixels created from the cubic convolution process. This method of interpolation does not require the samples used to be evenly spaced. The Akima A method uses a third order polynomial for interpolation. The interpolating polynomial is defined by the coordinates and the slopes of the two points that are on either side of the point to be interpolated. The slopes of the adjacent points are determined as follows:

If five points are defined as 1, 2, 3, 4, and 5 then the slope at point 3, t, is defined as:

$$t = \frac{|m_4 - m_3|m_2 + |m_2 - m_1|m_3}{|m_4 - m_3| + |m_2 - m_3|}$$

Where:

m_1 = slope of line segment defined by points 1 and 2
m_2 = slope of line segment defined by points 2 and 3
m_3 = slope of line segment defined by points 3 and 4
m_4 = slope of line segment defined by points 4 and 5

The Akima A method of interpolation is based upon the values (*y*) and slopes (*t*) on either side of the point that is to be interpolated. The interpolating polynomial for a point x between x_i and x_{i+1} is then defined as:

$$y = y_i + t_i * (x - x_i) + \frac{3\frac{y_{i+1} - y_i}{x_{i+1} - x_i} - 2t_i - t_{i+1}}{x_{i+1} - x_i} * (x - x_i)^2 + \frac{t_i + t_{i+1} - 2\frac{y_{i+1} - y_i}{x_{i+1} - x_i}}{(x_{i+1} - x_i)^2} * (x - x_i)^3$$

Where:

x = sample location of point to be interpolated
x_i = location of point to the left of x
x_{i+1} = location of point to the right of x
y_i = DN value for the input point at x_i
y = interpolated DN value for an output line and sample location

This methodology must be adjusted somewhat to account for higher frequency image distortion effects than those that can be captured by the conventional resampling grid. To model such effects, the LDCM attitude data stream is separated in to low-frequency and high-frequency segments with the low-frequency portion being used for the TIRS line-of-sight projection operations that build the resampling grid. The high-frequency data are interpolated to match the TIRS line sampling times and stored in the TIRS LOS model in a jitter table for application as an extra correction at image resampling time. The process of separating the attitude data stream by frequency is described in the TIRS Line-of-Sight Model Creation Algorithm Description Document.

Sensitivity coefficients that relate these high-frequency roll-pitch-yaw jitter terms to the equivalent input image space line and sample offset effects are stored in the TIRS LOS grid. This makes it possible to look up the roll-pitch-yaw jitter for each image line being resampled, and convert the jitter values to compensating input line/sample corrections that are used to refine the image interpolation location coordinates. The generation of these sensitivity coefficients is described in the TIRS Line-of-Sight Projection/Grid Generation Algorithm Description Document. The process by which the jitter table from the TIRS model and jitter sensitivity coefficients from the TIRS grid are used during image resampling is shown schematically in Figure 5 below. The items in green in the figure are new structures added to support jitter correction.

Since the jitter effects vary by image line, the time delay between deselected detectors can lead to slightly different jitter effects in adjacent image samples. This is depicted below in Figure 6. Six time samples (t0 through t5) for six adjacent detectors are shown in the figure. In this example, every other detector is assumed to be deselected and replaced from the redundant detector row. Note that the input line location returned by the grid is adjusted differently for the even and odd detectors due to their timing offset. Including the effects of detector deselect, the interpolated line location for the hybrid pixels could be different for each detector. The current approach does not account for sample-to-sample variations in jitter for each detector, applying the jitter correction only at the output location. This preserves the uniform along-track sampling assumption required to apply the cubic convolution kernel. Also note that while it is the interpolation location that is adjusted relative to the input pixel locations in the line direction, it is the detector sample locations that are adjusted relative to the interpolation location in the sample direction. The jitter-adjusted resampling procedure is explained in more detail below.

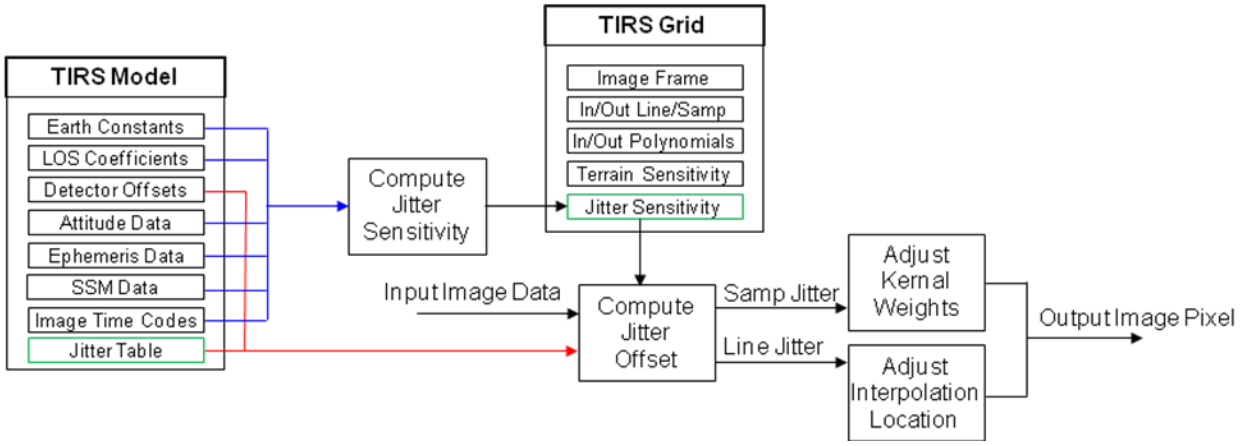


Figure 5: TIRS LOS Model and TIRS LOS Grid Jitter Correction Data Flow

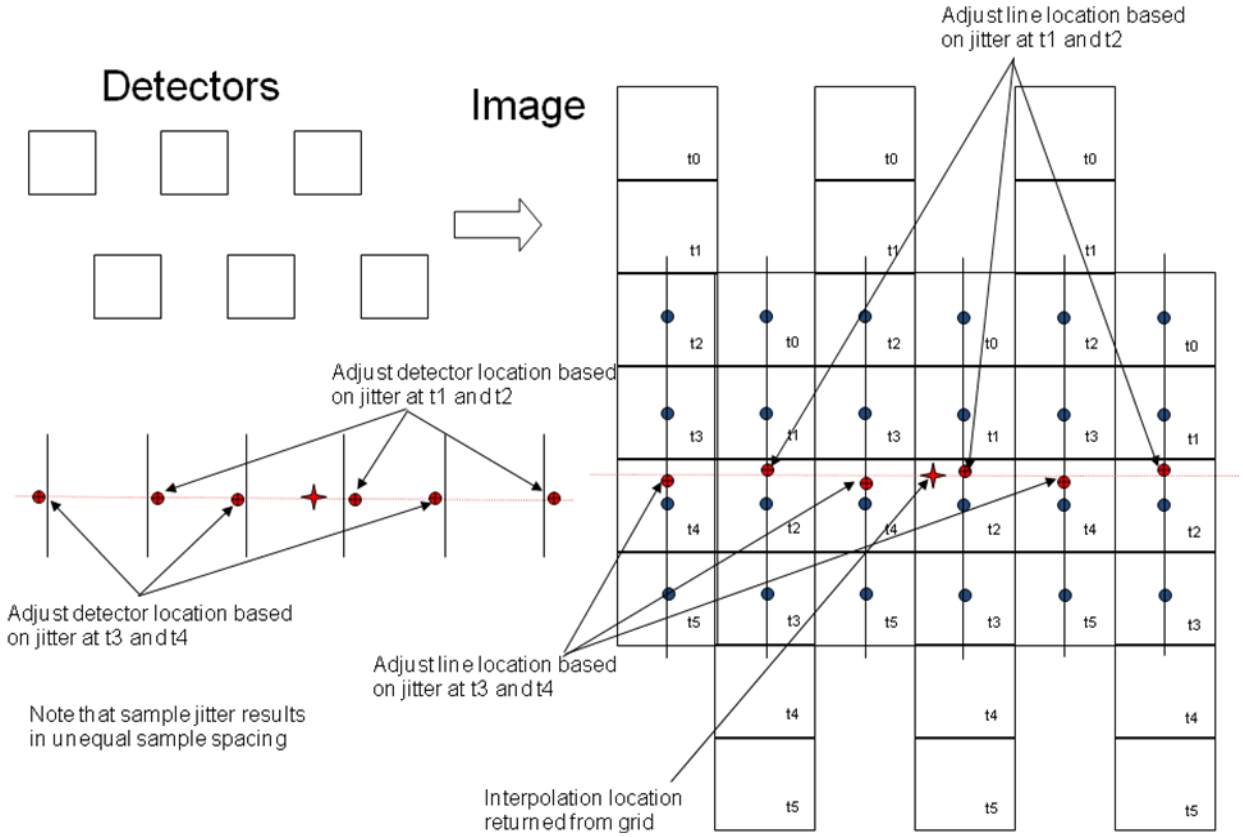


Figure 6: Jitter Effects in Image Resampling

7.3.3.6.1 Building The SCA-trimmed Look Up Table (LUT).

Allocate SCA-trim LUT. There is a starting and ending sample location of active or valid imagery stored for each line of output in the SCA-trimming look up table.

LUT = malloc(2 * nl)

Where nl = number of lines in output imagery

Given the set of geographic corner coordinates, read from the input grid file, that represent valid imagery for a given band:

7. Map four corners to output projection coordinates.
8. Map four output projection coordinates to line and sample coordinates.
9. Set up polygon definition from four coordinates:
   <px0,py0> = <sample upper left, line upper left>
   <px1,py1> = <sample upper right, line upper right>
   <px2,py2> = <sample lower right, line lower right>
   <px3,py3> = <sample lower left, line lower left>
   <px4,py4> = <sample upper left, line upper left>
10. Set up sample locations for each line that is outside active imagery:
    osamp1 = -1.0
    osamp2 = output number of samples
    for nn = 0 to 3
        if px[nn] < osamp1 then osamp1 = px[nn] – 1.0
        if px[nn] > osamp2 then osamp2 = px[nn] + 1.0
11. Initialize LUT values to fill for all output lines:
    For nn = 0 to (2 * number of output lines)
        LUT[nn] = 0
12. For nn = 0 to number of output lines (nn and current line are synonymous).
    6.1. Define line by sample locations calculated from 4 and current line
        <x0,y0> = <osamp1, nn>
        <x1,y1> = <osamp2, nn>
    6.2. Determine intersection between sides of polygon defined in 3 and line defined in 6.1
        Initialize number of intersections for current line:
            intersections = 0
        For nn = 0 to 3
            (Simple line intersection routine)
            xlk = x0 – x1
            ylk = y0 – y1
            xnm = px[nn] – px[nn+1]
            ynm = py[nn] – py[nn+1]
            xmk = px[nn+1] – x1
            ymk = py[nn -1] – y1
            det = xnm * ylk – ynm * xlk
            if (| det | <= TOL) lines are parallel, no intersection found.
            s = (xnm * ymk - ynm * xmk) / det
            t = (xlk * ymk - ylk * xmk) / det
            if(s<0.0 || s>1.0 || t<0.0 || t>1.0)
                no intersection found
            else
                intersection found, calculate point:
                xp[intersections] = x1 + xlk * s
                yp[intersections] = y1 + ylk * s
                intersections++
    6.3. If number of intersections from 6.2. is two then the current line has valid active imagery and the look up table values are these intersections and represent the start and stop of valid imagery. Store values in SCA-trim lookup table.
            if xp[0] > xp[1]

```
            LUT[ 2 * nn ]      = xp[1]
            LUT[ 2 * nn + 1] = xp[0]
      else
            LUT[ 2 * nn ]      = xp[0]
            LUT[ 2 * nn + 1] = xp[1]
```

(Note: If number of intersections is not two then current line has no valid active imagery and SCA-trim lookup table will contain points outside of imagery, fill will be used).

7.3.3.6.2 Load/Build Information

To resample a Level 1R data set, the image file, grid file, geometric model, and, if the effects terrain are to be removed, a DEM must be opened. See note #3.

7.3.3.6.3 Resample Level1R Imagery

Loop on each band of each SCA for resampling.

6. Get resampling grid for the band and SCA to be processed.
7. Build SCA-trimming table.
8. Read one band of imagery for one SCA.
   8.1. Initialize jitter correction parameters for this band (jitter_scale = 1 for TIRS)
9. Loop on output line/samples
   9.1. Check to see if output line/sample is within SCA-trimming bounds.
      if output sample > LUT[2 * output line] &&
         output sample < LUT[2 * output line + 1] then proceed
      else output pixel = fill
   9.2. If image is terrain corrected, calculate elevation dependent input line/sample location.
      4.2.1) Get elevation for output pixel location X/Y location from DEM (elevation). See note #3.
      4.2.2) Map the output line/sample back into input space using the grid and the function 3d_ols2ils.
   9.3. If image is not terrain corrected calculate zero elevation (ellipsoid surface) input line/sample location.
      4.3.1) Set elevation to zero
      4.3.2) Map the output line/sample back into input space using the grid and the function ols2ils.
   9.4. Calculate actual input sample location; for sample location (int)input sample calculated from either 4.2 or 4.3:
      4.4.1) Calculate detector offset parallax scale.
      Scale = (int) floor(detector along track offset + 0.5) (in geometric model). See note #4.
      4.4.2) Calculate sample detector parallax offset
      $\Delta sample\_oe = (d_0 + elevation * d_1) * scale$
      Note that $(d_0 + elevation * d_1)$ is the parallax (in pixels) per pixel of along track offset from the nominal detector location.
      Where:
      $d_{0,1}$ = detector sample parallax coefficients stored in the grid
      4.4.3) Get sample fractional offset
      fractional sample offset =
         detector across track offset (in geometric model)
      4.4.4) Calculate sample jitter adjustment
         4.4.4.1) Calculate the index into the jitter table for the current image line
         jit_index = (int)(jitter_scale*(input line – pixel column fill (defined below)))

Make sure jitter index is within the range of the jitter table. Set to the min or max value (whichever is closest) if it is outside the range.

4.4.4.2) Calculate the fractional jitter table index

Δjit_index = jitter_scale * input line – floor(jitter_scale * input line)

4.4.4.3) Calculate simple sample jitter adjustment

samp_jitter0 = samp_sens[0] * jitter_table[jit_index].roll
+ samp_sens[1] * jitter_table[jit_index].pitch
+ samp_sens[2] * jitter_table[jit_index].yaw
samp_jitter1 = samp_sens[0] * jitter_table[jit_index+1].roll
+ samp_sens[1] * jitter_table[jit_index+1].pitch
+ samp_sens[2] * jitter_table[jit_index+1].yaw
samp_jitter = samp_jitter0 * (1-Δjit_index) + samp_jitter1*Δjit_index
Where:
samp_sens[0] is the sample direction jitter roll sensitivity,
samp_sens[1] is the sample direction jitter pitch sensitivity,
samp_sens[2] is the sample direction jitter yaw sensitivity,
for the current grid cell, from the TIRS grid.
jitter_table[n] is the jitter table roll-pitch-yaw vector for row n,
from the TIRS model.

4.4.4.4) Refine the sample jitter to compensate for line jitter

line_jitter0 = line_sens[0] * jitter_table[jit_index].roll
+ line_sens[1] * jitter_table[jit_index].pitch
+ line_sens[2] * jitter_table[jit_index].yaw
line_jitter1 = line_sens[0] * jitter_table[jit_index+1].roll
+ line_sens[1] * jitter_table[jit_index+1].pitch
+ line_sens[2] * jitter_table[jit_index+1].yaw
line_jitter = line_jitter0 * (1-Δjit_index) + line_jitter1*Δjit_index
Where:
line_sens[0] is the line direction jitter roll sensitivity,
line_sens[1] is the line direction jitter pitch sensitivity,
line_sens[2] is the line direction jitter yaw sensitivity,
for the current grid cell, from the TIRS grid.
This is the error in the line coordinate used above, due to line jitter.
samp_rate =
samp_sens[0]*(jitter_table[jit_index+1].roll-jitter_table[jit_index].roll)
+ samp_sens[1]*(jitter_table[jit_index+1].pitch-jitter_table[jit_index].pitch)
+ samp_sens[2]*(jitter_table[jit_index+1].yaw-jitter_table[jit_index].yaw)
This is the rate of change of sample jitter with line coordinate.
samp_jitter += line_jitter*samp_rate
This is the sample jitter correction adjusted for the effects of line jitter.

4.4.5) actual input sample = input sample - Δsample_oe – samp_jitter - fractional sample offset (See note #5). These corrections are subtracted rather than added because what we are doing here is, rather than adjusting the input space interpolation location, computing the apparent location of the detector to the left of the interpolation location to make sure we have the correct range of samples to feed the interpolation logic. If the above adjustments

lead to the "actual input sample" being greater than (to the right of) the original input sample location, then we move our sample range one more sample to the left. We perform a similar calculation on the detector to the right of the input space interpolation location to make sure that we don't have to shift one more sample in that direction. See also the note in section 4.6.2 below.

9.5. Create fractional pixel shift for current input location:

Δline = input line - (int) input line

Δsample = input sample - (int) input sample

9.6. Create aligned samples for Akima resampling by applying cubic convolution weights in line direction.

9.6.1. Loop on actual input sample location:

For hybrid sample = (int) actual input sample - 2 to (int) actual input sample + 3 (Note #5. One extra hybrid sample created to left and right of minimum number of samples needed for Akima interpolation)

For NN resampling, only the two closest hybrid sample locations are calculated, that for (int) actual input sample and (int) actual input sample + 1.

9.6.1.1. Calculate line and hybrid sample detector offset parallax scale

scale = (int) floor(detector along track offset + 0.5) (in geometric model). See note #4.

9.6.1.2. Calculate detector offset, parallax, and jitter correction for hybrid detector.

9.6.1.2.1. Detector offset and parallax corrections.

Δline_oe = (c_0 + elevation * c_1) * scale + pixel column fill - nominal detector fill - at_offset[hybrid sample]

Δsample_oe = (d_0 + elevation * d_1) * scale

Where:

$c_{0,1}$ = detector line parallax coefficients stored in the grid

$d_{0,1}$ = detector sample parallax coefficients stored in the grid. Note that (c_0 + elevation * c_1) is the along-track parallax (in pixels) per pixel of along-track offset from the nominal detector location and (d_0 + elevation * d_1) is the across-track parallax (in pixels) per pixel of along-track offset from the nominal detector location.

9.6.1.2.2. Jitter correction

The sample jitter correction is calculated as described in section 3.3.4 above. The line jitter correction is calculated as follows:

jit_index = (int)(jitter_scale*(input line – pixel column fill))

Δjit_index = jitter_scale * input line – floor(jitter_scale * input line)

line_jitter0 = line_sens[0] * jitter_table[jit_index].roll
+ line_sens[1] * jitter_table[jit_index].pitch
+ line_sens[2] * jitter_table[jit_index].yaw

line_jitter1 = line_sens[0] * jitter_table[jit_index+1].roll
+ line_sens[1] * jitter_table[jit_index+1].pitch
+ line_sens[2] * jitter_table[jit_index+1].yaw

line_jitter = line_jitter0 * (1-Δjit_index) + line_jitter1*Δjit_index

Where:

line_sens[0] is the line direction jitter roll sensitivity,

line_sens[1] is the line direction jitter pitch sensitivity,

line_sens[2] is the line direction jitter yaw sensitivity,

for the current grid cell, from the TIRS grid.

This is the error in the line coordinate due to jitter.

line_rate =
line_sens[0]*(jitter_table[jit_index+1].roll-jitter_table[jit_index].roll)
+ line_sens[1]*(jitter_table[jit_index+1].pitch-jitter_table[jit_index].pitch)
+ line_sens[2]*(jitter_table[jit_index+1].yaw-jitter_table[jit_index].yaw)
This is the rate of change of line jitter with line coordinate.
line_jitter += line_jitter*line_rate
This is the line jitter correction adjusted for the second order effects of line jitter. Note the similarity to the sample correction described in 4.4.4.4.

9.6.1.3. Calculate new hybrid line location.
hybrid line = (int)floor(input line + ∆line_oe + line_jitter).
Note that in this case we add the corrections since we are adjusting the interpolation location.

9.6.1.4. Calculate new fractional hybrid line location.
Δhybrid line = input line + Δline_oe + line_jitter – hybrid line
If |Δhybrid line| > 1 then the integer line index must be adjusted and Δhybrid line brought back into the -1 < Δhybrid line < 1 range (see note #5).

9.6.1.5. For CC resampling, apply cubic convolution in line direction to hybrid sample line DNs.

9.6.1.5.1. Calculate cubic convolution weights. See note #2

$$w_{n+2} = \sum_{n=-1}^{2} f(n - \Delta \text{hybrid line})$$

Where *f* is equal to cubic convolution function.

9.6.1.5.2. Apply cubic convolution weights to L1R DNs.
hybrid line DN = $w_0 * h_0 + w_1 * h_1 + w_2 * h_2 + w_3 * h_3$
Where
w_0, w_1, w_2, w_3 = Cubic convolution weights for Δhybrid line.
h_0 = DN from L1R for hybrid sample, input line location - 1
h_1 = DN from L1R for hybrid sample, input line location
h_2 = DN from L1R for hybrid sample, input line location + 1
h_3 = DN from L1R for hybrid sample, input line location + 2

For NN resampling, use the fractional hybrid line location to select the closest integer line number, and extract the corresponding pixel value as the hybrid line DN for the current hybrid sample.

9.6.2. Calculate the apparent Akima pixel location for the current hybrid sample.
Akima pixel location x_i =
hybrid sample location - Δsample_oe - across track detector offset (in geometric model) – samp_jitter (computed per section 4.6.1.2.2 above).
Note that in this case the across-track terrain parallax and sample jitter effects are subtracted instead of added. This is because we are adjusting the apparent detector location relative to the output pixel interpolation point instead of adjusting the output pixel interpolation location itself. We must do it this way in the sample direction because the adjustments are different for each detector. As for the across-track offset term, which is also unique for each detector, the detector offset corrections are designed to be applied as line-of-sight corrections in the instrument coordinate system. As such, the along-track offset is a +X LOS correction and the across-track offset is a +Y LOS correction. The instrument +X axis is in the +line direction but

the +Y axis is in the –sample direction, so this correction is also subtracted from the apparent detector location.

9.7. For CC resampling, calculate output DN using Akima interpolation and hybrid line/sample information from 4.6.1 and 4.6.2.

9.7.1. Calculate Akima weights according to pixel locations from 4.6.2.

$$m_0 = \frac{\mathrm{DN}_1 - \mathrm{DN}_0}{x_1 - x_0}$$

$$m_1 = \frac{\mathrm{DN}_2 - \mathrm{DN}_1}{x_2 - x_1}$$

$$m_2 = \frac{\mathrm{DN}_3 - \mathrm{DN}_2}{x_3 - x_2}$$

$$m_3 = \frac{\mathrm{DN}_4 - \mathrm{DN}_3}{x_4 - x_3}$$

$$m_4 = \frac{\mathrm{DN}_5 - \mathrm{DN}_4}{x_5 - x_4}$$

$$ak_0 = \mathrm{DN}_2$$

$$ak_1 = \frac{|m_3 - m_2| * m_1 + |m_1 - m_0| * m_2}{|m_3 - m_2| + |m_1 - m_0|}$$

$$ak_2 = \frac{(3.0 * m_2 - 2.0 * ak_1 - \frac{|m_4 - m_3| * m_2 + |m_2 - m_1| * m_3}{|m_4 - m_3| + |m_2 - m_1|}}{x_3 - x_2}$$

$$ak_3 = \frac{ak_1 + \frac{|m_4 - m_3| * m_2 + |m_2 - m_1| * m_3}{|m_4 - m_3| + |m_2 - m_1|} - 2.0 * m_2}{(x_3 - x_2)^2}$$

Where:

DN_n = hybrid DNs calculated from cubic convolution, step 4.6.1.

x_n = Akima locations calculated in step 4.6.2.

ak_n = Akima weights

m_n = Akima slopes

9.7.2. Calculate output pixel DN using Akima A method.

$$\mathrm{output\,DN} = ak_0 + ak_1 * ds + ak_3 * ds^2 + ak_4 * ds^3$$

Where

$$ds = (\Delta\mathrm{sample} + x_2)$$

The output sample point is located between hybrid samples x_2 and x_3 where x_n is from n=0…5. For NN resampling, test the hybrid Akima locations for the extracted hybrid samples to decide which is closest to the desired output location. Select the closest hybrid sample value as the output DN.

9.8. Write output DN to image file. See note #7.

10. Write out data descriptor record for image file. The baseline contents of the metadata record are shown in Table 1. All fields present in the table refer to the imagery associated with the DDR unless otherwise specified. Note that the scene roll angle is a new field added for off-nadir

acquisitions. It would be computed from the TIRS LOS model by interpolating the roll angle from the "original" attitude data sequence at the time corresponding to the precision model reference time t_ref. This would be done using the logic described in the Find Attitude sub-algorithm in the TIRS LOS Projection ADD, except operating on the "original" rather than the "corrected" attitude data sequence. The logic for using the "original" data is so that this scene roll value will not change due to LOS model correction. The sign convention on the roll angle is such that a positive roll angle would correspond to a positive orbital Y coordinate which is looking to starboard.

7.3.3.6.4 Combining SCAs into one output file.

For an SCA combined output image the overlap region between SCAs can be handled by averaging the pixels between SCAs.

7.3.3.7 Prototype Code

Input to the executable is an ODL file, output is a HDF5 file containing the image data and corresponding metadata. The output format follows the format of the L1G DFCB version 1.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The following text is a brief description of the main set of modules used within the prototype with each module listed along with a very short description. It should be noted that not all library modules are referenced in the explanations below. The modules within the main directory of the prototype are discussed and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

Main driver for resampler (oliresample)

Main driver for TIRS resampler. Performs the following steps or calls the following modules.

1) Read input ODL parameters (tirs_getpar).
2) Read TIRS LOS model file (oli_get_model).
3) Read TIRS grid headers (oli_get_grid_headers).
If terrain correction read DEM file (oli_get_dem).
4) Open L1G image file (open_l1g_resamp_image).
5) Get fill pixel value (get_fill_pixel).
For each band to process
    6) Read grid band pointers (oli_get_grid_pointers).
    7) Open/initialize L1G band file (start_l1g_resamp_band).
    8) Setup resampling kernel (Kernal_Setup).
    9) Read resampling kernal information for resampling (get_kernal_info).
    For each SCA
        10) Read one SCAs worth of data from L0ra (get_input_image_data_l0ra).
        11) Resample SCA worth of data (resample_image).
        if not SCA combined image file write SCAs worth of data (write_l1g_resamp_band).
    If SCA combined image file write full SCA file (write_l1g_resamp_band).
    12) Close band in L1G output file (stop_l1g_resamp_writing_band).

13) Free grid band pointer (oli_free_grid).
14) Close L1G image file (close_l1g_resamp_image).
15) Update L1G metadata (update_l1g_metadata).

Get resampling processing parameters (tirs_getpar).
This function reads the TIRS resampling parameters from the ODL file. Also contains two functions, get_combine_sca and get_fill_pixel, that will return input flags as to whether 1) combine the SCAs in the output image and 2) what DN value should be used for fill.

Resample a given set of DN value using the Akima method (akima).
Function takes a given set of X locations with corresponding Y values and finds the Y value for the given input X location (xp). Function returns interpolated Y value associated with coordinate xp.

Calculate cubic convolution weight for a given location (cubic_convolution).
Given a cubic convolution alpha parameter and X value return the Y value associated with the cubic convolution function.

For a given band read one SCAs worth of L0R imagery (get_input_image_data_l0ra).
Given a L0rp file name, band number, and SCA number read a SCAs worth of data from L0rp file. Number of lines to read is taken from number of lines stored in models image data structure.

Set up resampling kernel (module kernal.c).
Using a set of functions, create a set of resampling weights. The resampling kernel is created and managed though several steps within the kernal.c file.

Kernal_Setup sets up kernal table or pointer. Allocates pointer and calls Create_Resampling_Kernal_1D to create a set of cubic convolution weights. Set is a look-up table of 1D cubic weights representing 1/64 of a shift in pixel locations.

Cleanup_Kernal frees up cubic convolution pointer.

Create_Resampling_Kernal_1D creates a set of one dimensional cubic convolution based on the input alpha parameter.

Get_Resample_Weight_Table_Ptr returns a pointer containing a set of 1D cubic convolution weights.

get_lines_in_kernal returns number of lines in resampling kernal.

get_samples_in_kernal returns number of samples in resampling kernal.

num_left_kernal_samples returns number of resampling weights to the "left" of the point that is to be interpolated.

num_right_kernal_sample returns number of resampling weights to the "right" of the point that is to be interpolated.

num_top_kernal_lines returns the number of lines "above" the point to be interpolated.

num_bottom_kernal_lines returns the number of lines "below" the point to be interpolated.

get_kernal_step_size returns the offset size in pixels between two sets of resampling weights.

get_kernal_info returns the number of steps (or number of sets of weights) within the resampling kernal, total number of sets of weights within the resampling table, width of resampling kernal, and height of resampling kernal.

Read DEM file (oli_get_dem).
Reads (Image Processing Element) IPE L1G file contain DEM data.

Open, close, write to L1G output image file (file output_image_data.c)
The file output_image_data.c contains several routines used for managing the output L1G file. Calls and functions are listed below.
open_l1g_resamp_image opens a L1G file.
start_l1g_resamp_band opens one band within an L1G file.
write_l1g_resamp_band writes image data to L1G file.
stop_l1g_resamp_writing_band closes band within L1G file.
close_l1g_resamp_image closed L1G file.

Resample one SCA for one band of L0Rp imagery (file resample_image.c)
The file resample.c contains several functions used in resampling imagery.

setup_trim_lut builds a lookup table that contains the starting and ending output pixel of valid imagery. Everything outside of this bounds will be set as fill

cleanup_trim_lut frees static buffer that contains SCA-trimming lookup table array.

get_kernal_info retrieves resampling weight table and corresponding characteristics.

setup_detector_offsets stores the detector offsets, along and across, level-0R fill, and nominal detector fill within arrays. Used by resample_image for applying detector offsets when resampling imagery.

resample_image is the main guts of the resampler. Takes the image data, DEM data if terrain corrected, grid band pointer, and TIRS model structure to resample one SCA or one band of imagery. Loops on output pixels mapping each output pixel location to an input location and resamples L0Rp (or L1R when it becomes available) using algorithm described in procedure section.

calc_jitter computes the sample and line direction jitter corrections for the current input line/sample location. This corrections are the adjustments to the input space interpolation location required to compensate for the high frequency jitter present at the time of observation.

calc_jitter_samp is a simplified version of calc_jitter that computes only the sample direction jitter correction. It is implemented as a separate function for processing efficiency because it is invoked more frequently than calc_jitter.

Update L1G metadata information (update_l1g_metadata).
Update L1G metadata according to projection information stored within resampling grid.

Write out ENVI header file (write_envi_hdr).
Writes out ENVI header file for image flat file that is written to disk. Only used for testing purposes.

Input and Output File Details Output is a L1G image file formatted according to the L1G DFCB. The output is a HDF5 file. The metadata associated with the output file is listed below. This table follows the metadata fields in version 1 of the LDCM Level-1 G DFCB. The metadata is split up into a file metadata and band metadata. For further information on this format see the L1G DFCB. Not all fields within the prototype metadata fields are filled in with valid data. Fields in which data is *not* correctly filled are indicated in italics (see notes #8 and #9).

Field	Description	Type
Spacecraft source	*Spacecraft associated with data record*	*char[32]*
Instrument source	*Imaging instrument (TIRS)*	*char[32]*
WRS path	WRS path number	integer
WRS row	WRS row number	integer
Capture direction	*Ascending or descending pass*	*char[64]*
Capture date	*Date imagery was acquired by instrument*	*char[11]*

Capture time	*Time of day imagery was acquired by instrument*	*char[8]*
Scene roll angle	*Roll angle (in degrees) at the scene center*	*float*
Correction type	*Raw, L1R, L1G, L1Gt, L1T*	*char[8]*
Acquisition type	*Earth, lunar, stellar*	*char[8]*
Projection Code	Map projection code	integer
Zone code	UTM zone code	integer
Datum code	Datum code for map projection	char[16]
Spheroid code	Earth model for map projection	integer
Projection units	Distance units	char[8]
Projection coefficients	Parameters needed by coordinate transformation package. For the prototype code projection transformations are performed using GCTP.	float[16]
For each band:		
Band Number	LDCM band designation for current record	integer
Number lines	Number of lines present in data file	integer
Number samples	Number of samples present in data file	integer
Data Type	Data type of imagery	integer
Maximum Pixel	Maximum DN value in data	float
Minimum Pixel	Minimum DN value in data	float
Maximum Radiance	Maximum radiance	float
Minimum Radiance	Minimum radiance	float
Upper left projection coordinate	Upper left y (latitude/northing) and x (longitude/easting) coordinate	float[2]
Upper right projection coordinate	Upper right y (latitude/northing) and x (longitude/easting) coordinate	float[2]
Lower right projection coordinate	Lower right y (latitude/northing) and x (longitude/easting) coordinate	float[2]
Lower left projection coordinate	Lower left y (latitude/northing) and x (longitude/easting) coordinate	float[2]
Projection distance y	Pixel size for y map coordinate	float
Projection distance x	Pixel size for x map coordinate	float

Table 1. Metadata Contents

7.3.3.8 Maturity

1. Since the OLI 3D grid approach is adopted for TIRS, the OLI code was reused with limited modifications.
2. The bad detector replacement approach used by TIRS, in which detectors from the redundant row are swapped for bad detectors in the primary row, is similar to the detector select capability used by the OLI. TIRS will use the same detector offset approach, in which the along track detector offsets are stored in the CPF with the whole pixel adjustment needed due to the detector selected and the small sub-pixel adjustment, capturing deviations from the nominal Legendre polynomial LOS model, that was present in the heritage ALI CPF detector offset

field. The fractional detector offset is separated from the detector select offset at times during processing.

3. The TIRS LOS model will not specifically address the problem of multiple terrain intersections. A terrain occlusion mask will be generated to identify obstructed OLI pixels (see note #1 below for additional details), and the TIRS grid structure should be compatible with the terrain occlusion algorithm but that algorithm will not be modified to accommodate TIRS.
4. The baseline (non-threaded) resampler implementation generates combined-SCA images by simply overwriting pixels present in multiple SCAs. Thus, the output image will contain pixel values from the highest-numbered SCA that views each pixel. The more sophisticated threaded resampler explicitly merges overlapping pixels, taking the average of pixels seen by two SCAs. Either method is acceptable but the latter approach is preferred.

7.3.3.9 Notes

Some additional background assumptions and notes include:

1. The new logic required to calculate the terrain occlusion mask (particularly for off-nadir scenes) is documented in a separate ADD but may be implemented as part of the OLI resampling software for processing efficiency. If the OLI and TIRS resampling implementations are converged, this logic would also be present for TIRS.
2. For implementation of the prototype code, the cubic convolution weights are stored within a table. The cubic convolution weights are stored as sets of 4 within the table with each set representing a 1/64 pixel shift.
3. For implementation of the prototype, the elevation for each output pixel are stored within a table (or image file) with each entry within the table representing the elevation of every pixel within the highest resolution of output imagery required.
4. Due to the detector select option aboard OLI and the TIRS bad detector replacement strategy, the detector parallax coefficients stored within the grid are in units of nominal pixels. To get the geometric effects of parallax and offset for a particular pixel the coefficients need to be rescaled to the pixel of interest based on its whole-pixel even/odd and deselect offset.
5. The processing needs to handle situations where fractional location shifts changes the indexing, or integer, location for the pixel of interest. This can happen when operating in either the sample or line direction.
6. The lunar L1Rs may contain detector offsets between primary and replaced detectors that are much greater in spacing than what is typically present in an earth imaged L1R. Since lunar processing is not required for TIRS, no special logic will be developed to handle this situation.
7. This assumes that the ALIAS heritage method of applying the radiance to product DN scaling within the resampler remains the same. Any scaling that needs to be done for the output imagery, whether due to output data type constraints or any radiometric scaling that must be applied, can be performed as the last step just before the DN value is written to the output image file.
8. Many of the fields in the band and file metadata could most likely be filled in from the L0Rp metadata. As this file format is still in the state of being defined these fields should be filled in as they become more finalized.
9. It currently is expected that the L0rp metadata file will contain the off-nadir angle associated with the image acquisition. Under this scenario this off-nadir angle from the L0rp can be used as the off-nadir angle within the resampled L1G metadata.

7.3.4 TIRS Band-to-Band Calibration Algorithm

7.3.4.1 Background/Introduction

The TIRS Band-to-Band calibration (B2BCal) algorithm estimates improved values for band placement within each Sensor Chip Assembly (SCA) of the TIRS instrument. Adjustments are made relative to the primary detector row 10.8 micrometer band, or in other words, the 10.8 micrometer band serves as the reference for all other bands. The baseline algorithm calibrates only the TIRS primary row 12.0 micrometer band relative to the primary row 10.8 micrometer band. Some features are retained from the OLI band calibration algorithm that would facilitate adding the capability to calibrate the TIRS redundant rows in the future.

The B2B calibration takes the TIRS Band Accuracy Assessment residuals file, which represents displacements with respect to the product output projection space, maps the residuals back into displacements with respect to the focal plane and then performs a least squares (LSQ) fit between the focal plane residuals to determine updates to the TIRS band Legendre line-of-sight (LOS) polynomial coefficients. The least squares fit results represent updates needed to adjust the existing Legendre LOS coefficients. These updates can be used to produce new Legendre LOS coefficients for the Calibration Parameter File (CPF).

TIRS band alignment calibration algorithm would be applied to the output of the TIRS band registration accuracy assessment algorithm derived from SCA-separated TIRS images.

7.3.4.2 Dependencies

The TIRS B2B calibration algorithm assumes that a cloud free nadir viewing L1T image has been generated and the resampled DEM used to create the L1T is available. The TIRS Model Creation and TIRS LOS Projection/Gridding algorithms for the L1T will be assumed to have been executed and the corresponding output files available. The L1T image needs to be in SCA-separated format and either in a SOM or UTM path oriented projection. The digital orthophoto quadrangle (DOQ) control and a digital elevation model (DEM) need to be used in generating the L1T. The accuracy of the precision solution should have post-fit residuals below the recommended threshold, the solution should have used an adequate number of control points, and the distribution of the control should be well distributed throughout the imagery. The TIRS Band Registration Accuracy Assessment (BRAA), or Band Characterization (B2BChar), algorithm will assumed to have been run on the L1T image successfully producing a B2B residuals file.

7.3.4.3 Inputs

The B2B calibration algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs	**Algorithm Inputs**
TIRS resampling grid	ODL
DEM	ODL
TIRS CPF file name	ODL
Along track IFOV	CPF/LOS-model (See note #8)
Minimum points	ODL
Number of Legendre Coefficients	ODL (See note #6)
TIRS Line-of-Sight model	ODL

B2B residuals file	ODL
Band calibration report file	ODL
Trend flag	ODL
Flag for CPF group creation (see note #3)	ODL
Flag for individual tie-point listing	ODL
CPF effective dates (begin and end)	ODL
Work Order ID (for trending)	ODL

7.3.4.4 Outputs

B2B calibration report file (See note #1 and table #1)
Legendre LOS CPF group
B2B calibration trending
Geometric Characterization ID
Work Order ID
WRS Path/Row
B2B calibration post and pre fit residuals
New SCA line-of-sight parameters

7.3.4.5 Options

Trending on/off switch

7.3.4.6 Prototype Code

The TIRS prototype code follows very closely that of the OLI prototype code. The changes and differences between the code are associated with the increase in the number of Legendre coefficients (from 3 to 4) and the decrease in the number of bands (10 to 4) and SCAs (14 to 3).

Input to the executable is an ODL file; output is an ASCII file containing measured offsets between band combinations of the L1T image and the corresponding updated line-of-sight (LOS) Legendre CPF coefficients. Under this directory is the ODL input file needed, band accuracy assessment residuals file, the input CPF, the output reports file and the output updated Legendre LOS coefficients.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The following are a set of brief descriptions of the main set of modules used within the prototype. It should be noted that almost all library modules are not referenced in the explanations below. The modules within the main bandcal directory or the prototype are discussed and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

getpar

Reads the parameters from the input ODL parameter file. Input parameters include: co-registered DEM, CPF, LOS resampling grid, geometric LOS model file and output band calibration report file names, the minimum points, number of coefficients, effective CPF file dates, and output file print flags. The minimum points variable ensures that the normal matrix contains a minimum number along its diagonal to zero out any omitted bands. Rather than being removed from the solution, the offsets for omitted bands are set to zero with a weight equal to the minimum number of points.

Omitted bands, for calibration or adjustment, are dependent on the bands present within the band accuracy assessment residuals file. A similar approach is used to restrict the number of SCAs that will be calibrated.

read_b2bchout
Reads band accuracy assessment residuals file. Also determines the specific SCAs and bands to calibrate by checking the band accuracy assessment residuals for SCAs and bands present (See Note #7).

oli_get_dem
Reads DEM file into IMAGE data structure.

oli_get_model
Reads TIRS geometric/LOS model.

oli_get_grid
Reads TIRS LOS geometric resampling grid.
19.
20. bandcal
21. Main driver for determining new Legendre LOS. Calls module to retrieve ODL parameters (getpar), calls module to read band accuracy assessment residuals files (read_b2bchout), reads elevation or DEM data (oli_get_dem), reads LOS geometric resampling grid, reads geometric line-of-sight model, and solves for new Legendre LOS (solve_focal_plane).
22.
23. solve_focal_plane
24. Module to solve for new Legendre LOS. Loops on each valid tie-point for each SCA and each band combination. Calls module get_los_errors to determine per tie-point adjustment needed for determining least squares (LSQ) solution for new Legendre LOS coefficients. Calculates post and pre-fit statistics associated with Legendre LOS coefficients.
25.
26. get_los_errors
27. Calculates delta input line and sample LOS needed for LSQ. Reads elevation for tie-point, maps tie-point to input space, finds adjustment needed between search and reference LOS vectors.
28.
29. write_SCA_parameters_cpf
Writes out a new set of Legendre LOS. Format fits LOS group within TIRS CPF.

7.3.4.7 Procedure

Band calibration uses the residuals measured during the TIRS Band Registration Accuracy Assessment Algorithm (See the TIRS Band Registration Accuracy Assessment ADD) to determine updates to the Legendre LOS coefficients (See TIRS Line-of-Sight Model Creation ADD). The band calibration process involves taking the residuals from band registration accuracy assessment, measured in output space, mapping them into input space angular deltas in terms of along- and across-track LOS angles and performing a least squares fit of the input space LOS angle deltas to a set of 3rd order Legendre polynomial correction coefficients. The correction polynomials calculated represent updates to the original LOS Legendre polynomial coefficients. New Legendre LOS coefficients can be found by combining the correction coefficients with the original coefficients.

Due to the differences in viewing geometry between bands within a SCA, along with the differences in viewing geometry between SCAs, the effects due to relief displacement must be taken into account during band calibration. To account for relief displacement during B2B calibration a DEM is required. The resampling grid and LOS model is also required during B2B calibration. The resampling grid, the corresponding detector's IFOV, and the LOS model's Legendre coefficients are used to map the residuals from output space to angular differences in input space.

A least squares fit is done on all requested bands and SCAs using the band-to-band tie point measurements from all band-pair combinations for a single SCA at a time. Requested bands and SCAs to process are based on the bands and SCAs present within the TIRS Band Registration Accuracy Assessment residuals file.

7.3.4.7.1 Stage 1- Data input

The data input stage involves loading the information required to perform the band calibration. Input file names are needed for: geometric LOS resampling grid, LOS model, band registration accuracy assessment results (B2B residuals file), output band calibration report file name, and the L1T DEM file name. Further input parameters are the effective begin and end dates of the new Legendre LOSs calculated, trending flag, CPF group creation flag, and individual tie-point reporting. Once the file names for the input data needed are retrieved the files can be opened and read.

Get ODL Parameters
Reads the parameters from the input ODL parameter file. This process was modified from the ALIAS heritage version to handle new inputs: minimum points, flag for CPF group creation, CPF effective dates, and flag for reporting individual tie-point results. The minimum points variable ensures that the normal matrix contains a minimum number along its diagonal to zero out any omitted bands. Rather than being removed from the solution, the offsets for omitted bands are set to zero with a weight equal to the minimum number of points.

Read Band-to-Band Residual File
Reads band accuracy assessment residuals file.

Read DEM
Read DEM file into IMAGE data structure.

Read TIRS LOS Model
Read TIRS geometric/LOS model.

Read TIRS LOS Geometric Grid
Read TIRS LOS resampling grid.

7.3.4.7.2 Stage 2 - Setup Least Squares Matrices and Solve

For each input SCA, every residual for each input band combination that is not an outlier is mapped back to TIRS input space. These input space mappings are single value adjustments needed for each point to align the LOS, associated with the focal plane, between the bands of the combination. This mapping procedure is described in more detail below. Once all of these residuals are mapped back to the focal plane and stored within the least-squares (LSQ) matrices new LOSs can be calculated.

The matrices defining calibration the process takes the following form:

$$[A][coeff\,] = [Y]$$

The matrices [*A*] and [*Y*] shown above correspond to one tie point measurement. The matrix [*coeff*] are the unknown adjustments to the Legendre LOS coefficients, the matrix [*A*] contain the Legendre coefficient multipliers for the band combination corresponding to that one measurement, and the [*Y*] matrix contains the input space residuals for that one measurement. For one measurement the matrices have the following dimensions:
[coeff] = (2 * Number of Legendre (4) * Number of bands (2)) x 1 = M x 1
[A] = 2 x (2 * Number of Legendre (4) * Number of bands (2)) = 2 x M
[Y] = 2 x 1

$$[coeff\,] = \begin{bmatrix} a_{b1,0} \\ a_{b1,1} \\ a_{b1,2} \\ a_{b1,3} \\ b_{b1,0} \\ b_{b1,1} \\ b_{b1,2} \\ b_{b1,3} \\ a_{b2,0} \\ a_{b2,1} \\ a_{b2,2} \\ a_{b2,3} \\ b_{b2,0} \\ b_{b2,1} \\ b_{b2,2} \\ b_{b2,3} \end{bmatrix}$$

Where:
$a_{bi,j}$ = Legendre coefficient j for line direction (along track) for band i
$b_{bi,j}$ = Legendre coefficient j for sample direction (across track) for band i
j = 0, 1, 2, 3 or the Number of Legendre coefficients to solve.
i = 1, 2 (Number of TIRS bands)

A 2x1 matrix pertaining to one residual measurement can be defined as:

$$[Y] = \begin{bmatrix} \Delta line \\ \Delta sample \end{bmatrix}$$

Where:

Δ*line* = input space residual in line direction (angular)
Δ*sample* = input space residual in sample direction (angular)

The TIRS input space residuals are calculated by finding the nominal (search) LOS in input space and the measured (search + measured offset) LOS in input space. These LOSs are found by mapping the output space line and sample locations to input space line and sample locations using the TIRS LOS projection grid (See TIRS Resampling ADD) and then using the TIRS LOS model (see TIRS Line-of-Sight Model Creation ADD) to convert the input space locations to LOSs. These input space nominal and measured locations are also used to construct the Legendre coefficient multipliers.

The design matrix [*A*] for one residual measurement is then:

$$[A_{nik}][coeff] = [Y_n]$$

$$[A] = \begin{bmatrix} -rl_{n,1,0} & -rl_{n,1,1} & -rl_{n,1,2} & -rl_{n,1,3} & 0 & 0 & 0 & 0 & sl_{n,2,0} & sl_{n,2,1} & sl_{n,2,2} & sl_{n,2,3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -rl_{n,1,0} & -rl_{n,1,1} & -rl_{n,1,2} & -rl_{n,1,3} & 0 & 0 & 0 & 0 & sl_{n,2,0} & sl_{n,2,1} & sl_{n,2,2} & sl_{n,2,3} \end{bmatrix}$$

Where:
$rl_{n,1,j}$ = reference band (1)1 Legendre polynomial
$sl_{n,2,j}$ = search band (2) Legendre polynomial
j = 0, 1, 2, 3 or the Number of Legendre coefficients to solve
n = tie-point number

These matrices define one observation. A sequence of observations can be summed to define the normal equations for a set of coefficients that can be used to update the TIRS LOS Legendre coefficients:

$$[N] = \sum A_{nik}^T W_{ik}^{-1} A_{nik}$$
$$[L] = \sum A_{nik}^T W_{ik}^{-1} Y_{nik}$$

Where [*N*] and [*L*] are summed over all n. Note that for TIRS there is only one band combination (the reference 10.8 micrometer band and the search 12.0 micrometer band). *W* is a weight matrix that is currently set to the same weight for all observations.

Since all of the tie point observations involve band differences, the solution lacks an absolute reference. To stabilize the solution a constraint observation is added to provide such a reference. This additional observation is applied to the 10.8 micrometer reference band as an offset of zero for each direction (line and sample). This fixes the reference band adjustment at zero and forces the search band to be registered to it.

$$[A_{00}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$[Y_{00}] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Where the 10.8 micrometer reference band is stored in the first eight columns of the [A] observation matrix.

The solution for a new set of Legendre coefficients is then:

$$[coeff] = [N]^{-1}[L]$$

Band Calibration Processing Steps

Note: Array indexes are zero-relative. Band numbers are 10.8 μm = 1, 12.0 μm = 2,
nLeg = Number of Legendre update coefficients to solve (1, 2, 3, 4 valid options).
Matrix indexes are zero relative

30. 1. Initialize parameters

31. $[W] = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$

Where σ^2 = 16, an approximate measurement variance for the tie point observations.

2. For each SCA to process

32. Initialize pre-fit statistics variables
33. pre-fit sum line = 0
34. pre-fit sum sample = 0
35. pre-fit sum line^2 = 0
36. pre-fit sum sample^2 = 0
37. Initialize LSQ matrices to zero
38. [N] = [0]
39. [L] = [0]

40.
41. 2.1 For the single TIRS band combination
42. rband = 10.8 μm reference band
43. sband = 12.0 μm search band
44.

2.1.1 For each tie-point

45. 2.1.2 Calculate reference line, sample location and search adjusted line, sample location.
46. rline = tie-point reference line location
47. rsamp = tie-point reference sample location
48. sline = tie-point search line location + line offset measured
49. ssamp = tie-point search sample location + sample offset measured
50. Note: sline, ssamp is the adjusted (or true) search location.
51.
52. Note that rline, rsamp, sline, ssamp are output space pixel locations.
53.
54. 2.1.3 Set rband and sband to zero-relative (needed for matrix operations, done with axx_parse_user_band in ALIAS)
55. rband = rband - 1
56. sband = sband - 1
57.
58. **Map residuals to input space (focal plane space).**

2.1.4 Find elevation for reference and sample locations

59. relev = elevation at rline,rsamp
60. selev = elevation at sline,ssamp
61.
62. 2.1.5 Map rline,rsamp and sline,ssamp to input space using 3d_ols2ils (See Note #2) and the search band TIRS LOS/resampling grid.
63. (riline,risamp) = 3d_ols2ils(search_grid, relev, rline, rsamp)
64. (siline,sisamp) = 3d_ols2ils(search_grid, selev, sline, ssamp)
65. Where
66. riline, risamp is the input space location of reference tie-point location.
67. siline, sisamp is the input space location of adjusted search tie-point location.
68. search_grid is the TIRS LOS/resampling grid for the search band.
69. Note: Search band grid is used for mapping both the adjusted search (siline,sisamp) and the reference locations.
70.
71. 2.1.6 Calculate Legendre normalized detector location
72.
$$\text{rnorm} = \frac{2.0 * \text{risamp}}{\text{number detectors in SCA} - 1} - 1$$

$$\text{snorm} = \frac{2.0 * \text{sisamp}}{\text{number detctors in SCA} - 1} - 1$$

73.
74. rnorm = normalized reference detector
75. snorm = normalized adjusted search detector
76.
77. 2.1.7 Calculate reference and search along and across track LOS.

$$nom\_sear\_x = coef\_x_{s,0} + coef\_x_{s,1} * rnorm + coef\_x_{s,2} * (3 * rnorm^2 - 1)/2 + coef\_x_{s,3} * rnorm * (5 * rnorm^2 - 3)/2 \quad (78.)$$

$$nom\_sear\_y = coef\_y_{s,0} + coef\_y_{s,1} * rnorm + coef\_y_{s,2} * (3 * rnorm^2 - 1)/2 + coef\_y_{s,3} * rnorm * (5 * rnorm^2 - 3)/2$$

$$sear\_x = coef\_x_{s,0} + coef\_x_{s,1} * snorm + coef\_x_{s,2} * (3 * snorm^2 - 1)/2 + coef\_x_{s,3} * snorm * (5 * snorm^2 - 3)/2 \quad (79.)$$

$$sear\_y = coef\_y_{s,0} + coef\_y_{s,1} * snorm + coef\_y_{s,2} * (3 * snorm^2 - 1)/2 + coef\_y_{s,3} * snorm * (5 * snorm^2 - 3)/2$$

Where
ref_x, ref_y = along and across track view angles
sear_x, sear_y = along and across track view angles
coef_x$_{s,n}$ = search Legendre along track coefficients
coef_y$_{s,n}$ = search Legendre across track coefficients

2.1.8 Determine LOS vectors
sear_z = 1.0

$$m = \sqrt{sear\_x^2 + sear\_y^2 + sear\_z^2}$$

$$sear\_x = \frac{sear\_x}{m}$$

$$sear\_y = \frac{sear\_y}{m}$$

$$sear\_z = \frac{sear\_z}{m}$$

nom_sear_z = 1.0

$$m = \sqrt{nom\_sear\_x^2 + nom\_sear\_y^2 + nom\_sear\_z^2}$$

$$nom\_sear\_x = \frac{nom\_sear\_x}{m}$$

$$nom\_sear\_y = \frac{nom\_sear\_y}{m}$$

$$nom\_sear\_z = \frac{nom\_sear\_z}{m}$$

2.1.9.1 Determine effective line-of-sight instantaneous-field-of-view (IFOV)

2.1.9.1.1 Map input search pixel location, line and sample, to output space.
sline = search line location
ssamp = search sample location
elevation = elevation for location sline, ssamp

Calculate elevation planes bounding current elevation.

$$zplane = \frac{elevation}{grid\ z\ spacing} + grid\ zero\ plane$$

*elev0 = grid z spacing * (zplane – grid zero plane)*
elev1 = elev0 + grid z spacing

Calculate cell index, row and column, for search line and sample location and zplane.
row = sline / grid cell line spacing
column = ssamp / grid cell sample spacing
*cell index0 = nrows * ncols * zplane + row * ncols + column*

Where:
grid z spacing = elevation difference between two grid planes
ncols = number of grid cell columns
nrows = number of grid cell rows

Calculate output space line, sample location for input space search line, sample location and zplane.
$a_{0,1,2,3}$ = grid sample location forward mapping coefficients for cell index0
$b_{0,1,2,3}$ = grid line location forward mapping coefficients for cell index0

*lms = sline * ssamp*
*osamp0 = a_0 + a_1 * ssamp + a_2 * sline + a_3 *lms*
*oline0 = b_0 + b_1 * ssamp + b_2 * sline + b_3 * lms*

Calculate cell index, row and column, for search line and sample location and zplane +1.

*cell index1 = nrows * ncols * (zplane + 1.0) + row * ncols + column*

Calculate output space line, sample location for input space search line, sample location and zplane+1.

$a_{0,1,2,3}$ = grid sample location forward mapping coefficients for cell index1
$b_{0,1,2,3}$ = grid line location forward mapping coefficients for cell index1

*lms = sline * ssamp*
*osamp1 = a_0 + a_1 * ssamp + a_2 * sline + a_3 *lms*
*oline1 = b_0 + b_1 * ssamp + b_2 * sline + b_3 * lms*

Calculate output space line, sample location for input space search line, sample location, and elevation.

w0 = (elev1 – elevation) / (elev1 – elev2)
w1 = (elevation – elev0) / (elev1 – elev2)

*$osamp_n$ = osamp0 * w0 + osamp1 * w1*
*$oline_n$ = oline0 * w0 + oline1 * w1*

2.1.9.1.2 Map input location ssamp, sline+1.0 to output space $osamp_{n+1}$,$oline_{n+1}$ (repeat step 2.1.9.1.1 for input location ssamp,sline+1)

2.1.9.1.3 Determine change in output space between input locations (ssamp, sline) and (ssamp, sline+1.0)
dline = $oline_n$ – $oline_{n+1}$
dsamp = $osamp_n$ = $osamp_{n+1}$

$$\text{distance} = \sqrt{\text{dline} * \text{dline} + \text{dsamp} * \text{dsamp}}$$

2.1.9.1.4
If earth acquisition calculate LOS distance to target

Calculate output latitude and longitude for search line and sample (see Forward Model section of LOS Projection Ellipsoid & Terrain).

Calculate time for current search line and sample (see section a.1 in Forward Model section of LOS Projection Ellipsoid & Terrain).

Calculate satellite position for current search line and sample time (see section a.4 in Forward Model section of LOS Projection Ellipsoid & Terrain).

Calculate target vector (see section a.7 in Forward Model section of LOS Projection Ellipsoid & Terrain).

LOS x coordinate = target x coordinate - satellite x coordinate
LOS y coordinate = target y coordinate - satellite y coordinate
LOS z coordinate = target z coordinate - satellite z coordinate

length = sqrt(LOS x * LOS x + LOS y * LOS y + LOS z * LOS z)

$IFOV_{along}$ = (output pixel size * distance) / length

If lunar acquisition

$IFOV_{along}$ = output pixel size * distance

2.1.10 Calculate the LOS errors

$$\Delta line = \frac{sear\_x}{sear\_z} - \frac{nom\_sear\_x}{nom\_sear\_z} + (siline - riline) * IFOV_{along}$$

$$\Delta samp = \frac{sear\_y}{sear\_z} - \frac{nom\_sear\_y}{nom\_sear\_z}$$

2.1.11 Create matrices need to sum with [N] and [L].

2.1.11.1 Calculate Legendre generating polynomial coefficients for search and reference:
80. sl_0 = 1.0
81. if(*nLeg* >= 2) sl_1 = snorm
82. if(*nLeg* >= 3) sl_2 = (3 * snorm^2 – 1)/2
83. if(nLeg == 4) sl_3 = snorm * (5 * snorm^2 – 3)/2

84. $rl_0 = 1.0$
85. if (*nLeg* >= 2) rl_1 = rnorm
86. if(*nLeg* >= 3) $rl_2 = (3 * \text{rnorm}^2 - 1)/2$
87. if(nLeg == 4) $rl_3 = \text{rnorm} * (5 * \text{rnorm}^2 - 3)/2$
88. Note: If the number of Legendre coefficients in the solution is less than 4 the corresponding sl_n and rl_n will be omitted.
89.
90. 2.1.11.2 Initialize [A] to zero and then set [A] indexes to sl_n and rl_n.
91. A[0][Number Legendre * sband + n] = sl_n
92. A[1][Number Legendre * sband + n] = sl_n
93. A[0][Number Legendre * rband + n] = $-rl_n$
94. A[1][Number Legendre * rband + n] = $-rl_n$
95. Where: n = 0 … *nLeg* -1
96. [A] = 0 elsewhere
97.
98. 2.1.11.3 Set [Y] according to input space deltas measured and sum pre-fit statistics.
99.
100. 2.1.11.3.1 Store deltas in [Y]
101. Y[0][0] = Δ*line*
102. Y[1][0] = Δ*samp*
103.
104. 2.1.11.3.2 Sum statistics
105. pre-fit sum line = pre-fit sum line + Δ*line*
106. pre-fit sum sample = pre-fit sum sample + Δ*sample*
107. pre-fit sum line^2 = pre-fit sum line^2 + Δ*line*^{2}
108. pre-fit sum sample^2 = pre-fit sum sample^2 + Δ*sample*^{2}
109.
110. 2.1.11.4 Create matrices to add to normal matrices
111. $[A_{\text{tie-point}}] = [A]^T[W][A]$
112. $[Y_{\text{tie-point}}] = [A]^T[W][Y]$
113.
114. 2.1.11.5 Sum N and L matrices
115. $[N] = [N] + [A_{\text{tie-point}}]$
116. $[L] = [L] + [Y_{\text{tie-point}}]$
117.
118. 2.2 Set minimum points for bands to omit from processing.
119. Eliminate observations for omitted band:
120. oband = band to omit - 1 (from earlier, bands are 1-relative)
121. $[N]_{g+n,i} = 0$
122. Where g = *nLeg* * 2 * oband
123. n = 0 … 2**nLeg* - 1
124. i = 0 … *nLeg* * 2 * Number of Bands - 1
125. $[N]_{i,g+n} = 0$
126. Where g = *nLeg* * 2 * oband
127. n = 0 … 2**nLeg* - 1
128. i = 0 … *nLeg* * 2 * Number of Bands - 1
129. $[N]_{g+n,g+n}$ = Minimum Points
130. Where g = *nLeg* * 2 * oband
131. n = 0 … 2**nLeg* - 1

132. $[L]_{g+n} = 0$
133. Where g = *nLeg* * 2 * oband
134. n = 0 … 2**nLeg* - 1
135.
136. 2.3 Solve for delta Legendre coefficients
137. $[\Delta coeff] = [N]^{-1}[L]$
138.
139. 2.4 Calculate new Legendre coefficients
140. $\text{new along coefficients}_{SCA,band,n} = \text{previous along coefficients}_{SCA,band,n} + \Delta coeff_{i+n}$
141. $\text{new across coefficients}_{SCA,band,n} = \text{previous across coefficients}_{SCA,band,n} + \Delta coeff_{i+nLeg+n}$
142. band = 1, 2
143. i = 2 * *nLeg* * (band – 1)n = 0 … *nLeg* - 1Stage 3 - Calculate Pre and Post fit Residuals

144. 1. For each SCA calculate residuals
145. Initialize post-fit statistics variables
146. post-fit sum line = 0
147. post-fit sum sample = 0
148. post-fit sumsq line = 0
149. post-fit sumsq sample = 0
150.
151. 2. For the single band combination
152.
153. 2.1 Perform steps 2.1.1 - 2.1.10 from stage 3.
154.
155. 2.2 Calculate adjusted reference and search line/sample locations
156. riline' = $\Delta coeff_{sca,rband,0}$
157. if(*nLeg* >=2) riline' = riline' + rnorm * $\Delta coeff_{sca,rband,1}$
158. if(*nLeg* >= 3) riline' = riline' + (3 * rnorm^2 - 1)/2 * $\Delta coeff_{sca,rband,2}$
159. if(*nLeg* == 4) riline' = riline' + rnorm * (5 * rnorm^2 - 3)/2 * $\Delta coeff_{sca,rband,3}$
160. risamp' = $\Delta coeff_{sca,rband,0}$
161. if(*nLeg* >= 2) risamp' = risamp' + rnorm * $\Delta coeff_{sca,rband,1}$
162. if(*nLeg* >= 3) risamp' = risamp' + (3 * rnorm^2 - 1)/2 * $\Delta coeff_{sca,rband,2}$
163. if(*nLeg* == 4) risamp' = risamp' + rnorm * (5 * rnorm^2 - 3)/2 * $\Delta coeff_{sca,rband,3}$
164. siline' = $\Delta coeff_{sca,rband,0}$
165. if(*nLeg* >= 2) siline' = siline' + snorm * $\Delta coeff_{sca,sband,1}$
166. if(*nLeg* >= 3) siline' = siline' + (3 * snorm^2 - 1)/2 * $\Delta coeff_{sca,sband,2}$
167. if(*nLeg* == 4) siline' = siline' + snorm * (5 * snorm^2 - 3)/2 * $\Delta coeff_{sca,sband,3}$
168. sisamp' = $\Delta coeff_{sca,sband,0}$
169. if(*nLeg* >= 2) sisamp' = sisamp ' + snorm * $\Delta coeff_{sca,sband,1}$
170. if(*nLeg* >= 3) sisamp' = sisamp' + (3 * snorm^2 - 1)/2 * $\Delta coeff_{sca,sband,2}$
171. if(*nLeg* == 4) sisamp' = sisamp' + snorm * (5 * snorm^2 - 3)/2 * $\Delta coeff_{sca,sband,3}$
172. Where:
173. SCA, band, 0, 1, 2, 3 are the SCA, band number and coefficients for the updates to the Legendre polynomials. The $\Delta coeff$ added to the riline are the along track updates the $\Delta coeff$ add to the risamp are the across track updates.

174. rband = index to reference band coefficient
175. sband = index to search band coefficient
176.
177. 2.3 Calculate new post fit Δ*errors* by updating Δ*line* and Δ*sample* with Legendre updates
178. Δ*line*' = Δ*line* - riline' + siline'
179. Δ*samp*' = Δ*samp* - risamp' + sisamp'
180. Where:
181. Δ*line* and Δ*samp* are the same as those calculated in 2.1.10 from stage 3.
182.
183. 2.4 Sum post-fit variables
184. post-fit sum line = post-fit sum line + Δ*line*'
185. post-fit sum sample = post-fit sum sample + Δ*sample*'
186. post-fit sumsq line = post-fit sumsq line + Δ*line*'^{2}
187. post-fit sumsq sample = post-fit sumsq sample + Δ*sample*'^{2}
188.
189. 3. Calculate post and pre fit statistics for both line and sample directions:

$$\text{mean} = \frac{\text{sum}}{\text{number of points}}$$

$$\text{scale} = \frac{1}{\text{number of points} * (\text{number of points} - 1)}$$

190.

$$\text{standard deviation} = \sqrt{\frac{\text{sum squares}}{\text{number of points} - 1} - \text{sum} * \text{sum} * \text{scale}}$$

$$\text{rmse} = \sqrt{\frac{\text{sum squares}}{\text{number of points}}}$$

191. Where
192. sum = pre/post sum line or pre/post sum sample
193. sum squares = pre/post sumsq line or pre/post sumsq sample
194. number of points = number of points used in LSQ fit
195.
196. 4. Create Band-to-Band Calibration output report (See table #1).
197. 4.1 Write report header information.
198.
199. 4.2. Write post and pre-fit statistics (per SCA) for line and sample direction.
200.
201. 4.3. Write individual tie-point statistics (if tie-point reporting flag = Yes).
202.
203. 5. If CPF group flag is set to yes write out ASCII file of CPF group with new Legendre coefficients.
204.

7.3.4.8 Output files

The output report contains a standard header. This standard header is at the beginning of the file and contains the following:

1) Date and time file was created.
2) Spacecraft and instrument pertaining to measurements.
3) Pointing (roll) angle of spacecraft/instrument.

4) Acquisition type
5) Report type (band-to-band)
6) Work order ID of process (left blank if not applicable)
7) WRS path/row
8) Software version that produced report.
9) L0R image file name

The following items should be stored (trended) in the database with respect to the Band-to-Band Calibration algorithm:

All report header information:
- Date and time
- Spacecraft instrument source
- Work order ID
- WRS path/row
- Software version
- Off-nadir angle
- L0Rp file name
- Processing file name

The following processing parameters:
- Bands processed
- SCAs processed

The following report file information:
- Number of points used per SCA
- Computed Legendre along track coefficient updates
- Computed Legendre across track coefficient updates
- New Legendre along track coefficients (updates + existing)
- New Legendre across track coefficients (updates + existing)
- Post-fit mean, standard deviation, RMSE
- Pre-fit mean, standard deviation, RMSE

See note #11.

465. Field	**466. Description**	**467. Trend**
468. Date and time	469. Date (day of week, month, day of month, year) and time of file creation.	470. Yes
471. Spacecraft and instrument source	472. LDCM and TIRS	473. Yes
474. Processing Center	475. EROS Data Center SVT	476. No
477. Work order ID	478. Work order ID associated with processing (blank if not applicable)	479. Yes
480. WRS path/row	481. WRS path and row (See note #4)	482. Yes
483. Software version	484. Software version used to create report	485. Yes
486. Off-nadir angle	487. Off-nadir pointing angle of processed image file (See note #5)	488. Yes
489. Acquisition Type	490. Earth viewing or Lunar	491. Yes
492. L0Rp image file	493. L0Rp image file name used to create L1T	494. Yes
495. Processed image file name	496. Name of L1T used to create report	497. Yes

498. Number of Legendre coefficients	499. Number of Legendre coefficients present	500. Yes
501. Heading for pre and post fit statistics	502. One line of ASCII text defining pre and post statistics	503.
504. For each SCA (along and across track directions)	505.	506.
507. SCA number	508. SCA number associated with statistics	509. Yes
510. Pre fit statistics	511. Mean, RMSE, standard deviation, along and across track direction (in units of radians)	512. Yes
513. Post fit statistics	514. Mean, RMSE, standard deviation, along and across track direction (in units of radians)	515. Yes
516. For each SCA and band	517.	518.
519. Along track solution	520. Legendre along track correction coefficients	521. Yes
522. Across track solution	523. Legendre across track correction coefficients	524. Yes
525. For each SCA and band	526.	527.
528. Along track updates	529. Updated Legendre along track coefficients	530. Yes
531. Across track updates	532. Updated Legendre across track coefficients	533. Yes
534. For each tie-point of each SCA and Band to process	535. Output produced only if tie-point results flag is set to Yes.	536.
537. Point ID	538. Point identifier	539. No
540. SCA number	541. SCA number for band combination	542. No
543. Reference output line	544. Output tie-point location in line direction	545. No
546. Reference output sample	547. Output tie-point location in sample direction	548. No
549. Reference input line	550. Reference input tie-point location in line direction	551. No
552. Reference input sample	553. Reference input tie-point location in sample direction	554. No
555. Search input line	556. Search input tie-point location in line direction	557. No
558. Search input sample	559. Search input tie-point location in sample direction	560. No
561. Reference band	562. Reference band	563. No
564. Search Band	565. Search band	566.
567. Measured line offset	568. Output space offset in line direction (from Band Accuracy Assessment residuals file)	569. No
570. Measured sample offset	571. Output space offset in sample direction (from Band Accuracy Assessment residuals file)	572. No
573. Pre-fit line delta	574. Pre-fit input space line delta/offset ($\Delta line$)	575. No
576. Pre-fit sample delta	577. Pre-fit input space sample delta/offset ($\Delta sample$)	578. No
579. Post-fit line delta	580. Post-fit input space line delta/offset ($\Delta line'$)	581. No
582. Post-fit sample delta	583. Post-fit input space sample delta/offset ($\Delta sample'$)	584. No

Table 1. Band Calibration Report file

If the CPF group creation flag is set to yes an ASCII file containing the updated Legendre LOS should be generated. This file would contain the new Legendre LOS for each SCA for every band and would be formatted according to the CPF group that the Legendre LOS resides in (for geometric prototype code this is the LOS_LEGENDRE group). The file would also contain the file attributes CPF group with the effective dates for the LOS generated (See note #3). The SCAs and bands that were not updated should still be represented within the file; these values should be the same for post and pre calibration.

7.3.4.9 Maturity

3. Band-to-Band Calibration for TIRS closely follows that of ALIAS and OLI.

7.3.4.10 Notes

Some additional background assumptions and notes include:

7. The band calibration results currently contains the L1T name, pre and post fit mean, root mean squared error, and standard deviation for the along and across track direction of each SCA, new Legendre LOS coefficients, and a new CPF Legendre LOS group parameters. The individual tie-point characteristic information and (pre and post-fit) residuals and should be added to the report file (see table #1).
8. See "Using the LOS geometric resampling grid to map an output pixel location to an input pixel location" in the TIRS Resampling ADD for ols2ils functionality.
9. The table listed below contains the file attributes and LOS groups that should be populated with the corresponding TIRS fields when the CPF group creation flag is set to yes. The CPF_Status, CPF_Name_Source, CPF_Description, and CPF_Version fields were inserted during ALIAS development by software development, these fields may or may not be present/needed for TIRS processing.

Parameter Groups	Parameter Name	Data Type	Description
GROUP: LOS_LEGENDRE	Along_LOS_Legendre_B***BB***_***NNN***_SCA***SS***	float32 array (4 values) for each band of each SCA	Legendre polynomial coefficients defining along track viewing angle of band number ***BB***, band name ***NNN*** and SCA ***SS*** given in radians Valid format: for each term: SN.NNNNESN, where S = "+" or "-", N = 0 to 9, and E = "E".
GROUP: LOS_LEGENDRE	Across_LOS_Legendre_B***BB***_***NNN***_SCA***SS***	float32 array (4 values) for each band of each SCA	Legendre polynomial coefficients defining across track viewing angle of band number ***BB***, band name ***NNN*** and SCA ***SS*** given in radians Valid format: for each term: SN.NNNNESN, where S = "+" or "-", N = 0 to 9, and E = "E"

The file name for the CPF group can follow the convention of:
Legendre_coefficients_<effective begin date>_<effective end date>.odl
Where:
effective begin date = YYYYMMDD

effective end date = YYYYMMDD
YYYY = Year
MM = Month of year
DD = Day of month

10. Any kind of "non-WRS" collect; lunar or off-nadir viewing at the poles should have 000/000 listed as the path/row.
11. Pointing angle for lunar acquisitions would be 0.0.
12. Currently it is not expected that any calibration will be done on anything other than the full range of Legendre coefficients (4), however the prototype code supports the range of 1-4 Legendre coefficients in the solution. The IAS prototype code will keep this option and it should remain in the system.
13. The normal operating procedure will be to calibrate all of the TIRS SCAs. Input that does not contain the full three TIRS SCAs is more of a toolkit and testing capability.
14. The IFOV was historically used for calculating the LOS errors of the measured residuals. The code was modified to calculate a dynamic IFOV, however the CPF static IFOV was left as an input to allow for comparisons between the historical and current dynamic methods.

7.3.5 TIRS Alignment Calibration Algorithm

7.3.5.1 Background/Introduction

The TIRS alignment calibration algorithm combines the functions of the OLI sensor alignment and focal plane alignment calibration algorithms. Using an OLI short-wave infrared (SWIR) band image as a reference it compares an SCA-separated precision and terrain corrected (L1T) TIRS 10.8 micrometer band image (see note #1) with the OLI SWIR reference image. Each SCA in the TIRS L1T image is compared to the SCA-combined OLI reference to measure both systematic full-scene TIRS-to-OLI misregistration and SCA-specific deviations from the scene-average registration. The measured deviations are used to estimate corrections to the TIRS-to-OLI alignment matrix and to the 10.8 micrometer band Legendre polynomial coefficients that model the nominal lines-of-sight for each SCA.

The algorithm is implemented in two steps: 1) a mensuration/setup step in which the separated-SCA L1T image is correlated with the OLI reference image to measure the within-SCA deviations, and; 2) a calibration update computation step in which the measured deviations are used to compute TIRS-to-OLI alignment corrections and TIRS line-of-sight model correction Legendre coefficients that adjust the original LOS model to minimize the residual image deviations. The calibration update step includes applying an outlier filter to the image measurements. Separating the algorithm into two distinct steps makes it possible to run the calibration update step multiple times, using different outlier filter thresholds, for example, without having to perform the time consuming image mensuration/correlation setup procedure more than once.

Results from individual calibration scenes are stored in the geometric trending database so that results from multiple scenes can be analyzed together when deciding whether and how to adjust the operational ACS-to-TIRS alignment and TIRS focal plane calibrations. If a 10.8 micrometer band focal plane calibration update is generated, the TIRS 12.0 micrometer spectral band would subsequently be re-registered to the 10.8 micrometer band using the TIRS band alignment calibration procedure.

The TIRS alignment calibration procedure is derived from the OLI focal plane calibration algorithm. The implementation should be very similar for the setup step, which measures the SCA-specific deviations relative to the reference image. The legendre step, which calculates the Legendre polynomial coefficient updates, will be enhanced to include the computation of the full-scene TIRS-to-OLI alignment update. Since the Legendre coefficients and alignment angles are not completely independent, some additional constraints are required to make the parameters separable. The default approach is to constrain the Legendre coefficients so that they cannot model roll, pitch, or yaw effects (more about this below). An alternate option is to constrain the alignment angles. This option makes the calibration solution mimic the heritage focal plane calibration procedure.

7.3.5.2 Dependencies

The TIRS alignment calibration algorithm assumes that the L1T process flow has created a substantially cloud-free SCA-separated (nadir-viewing) path-oriented L1T 10.8 micrometer band image, over a band registration calibration site, which has been registered to an OLI reference image either by using systematic LOS models for both the OLI and TIRS images, or by transferring the OLI-derived precision correction model to the TIRS LOS model. This would be accomplished by copying the OLI precision correction parameters from the OLI LOS model to the TIRS LOS model. The SCA-separated TIRS L1T image will be framed to exactly match the SCA-combined OLI reference image, by using the TRANSFER_FRAME framing option during TIRS LOS projection grid generation. This algorithm also assumes that the CPF, TIRS LOS model, TIRS grid file, and DEM used to produce the TIRS L1T image, are available.

7.3.5.3 Inputs

The TIRS alignment calibration algorithm uses the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data). The second column shows which algorithm step (image mensuration or correction model computation) uses the input.

Algorithm Inputs	**Processing Step**
ODL File (implementation)	Both
Calibration Parameter File (CPF) Name	Both
TIRS L1T Image File Name (see note #2)	Step 1
TIRS LOS Model File Name	Step 1
TIRS LOS Projection Grid File Name	Step 1
DEM File Name	Step 1
OLI Reference Image File Name	Step 1
Correlation Data File Name	Both
Report File Name	Step 2
Processing Parameters	Both
Number of Tie Points per Cell	Step 1
Outlier tolerance	Step 2
Constraint Type: 0 (default) = constrain Legendre coefficients (solve all parameters), 1 = constrain angles (solve Legendre only)	Step 2
Work order ID (for trending)	Step 2
WRS Path (for trending)	Step 2
WRS Row (for trending)	Step 2
Calibration effective dates for updated parameters	Step 2
Trending flag	Step 2
CPF	Both
Calibration effective dates	Step 2

ACS-to-OLI alignment matrix (3x3 orientation matrix)	Step 2
ACS-to-TIRS alignment matrix (3x3 orientation matrix)	Step 2
Algorithm Parameters (formerly system table parameters)	
Size of Correlation Window	Step 1
Peak Fit Method	Step 1
Min Correlation Strength	Step 1
Max Correlation Displacement	Step 1
Fill Threshold Fraction (max percent of window containing fill value)	Step 1
Tie point weight (in units of 1/microradians^2)	Step 2
Alignment constraint weight (in units of 1/microradians^2) (new)	Step 2
Fit order	Step 2
Post-fit RMSE Thresholds (trending metrics) (in units of microradians)	Step 2
TIRS Grid File (see LOS Projection ADD for details)	Step 1
Number of SCAs	Step 1
For each SCA:	Step 1
Grid cell size in lines/samples	Step 1
Number of lines/samples in grid	Step 1
Number of z-planes, zero z-plane index, z-plane spacing	Step 1
Array of grid input line/sample locations	Step 1
Array of output line/sample locations (per z-plane)	Step 1
Array of forward mapping coefficients	Step 1
Array of inverse mapping coefficients	Step 1
Rough mapping polynomial coefficients	Step 1
TIRS LOS Model File	Step 1
TIRS Along-Track IFOV (in radians)	Step 1
Number of SCAs	Step 1
Number of Bands	Step 1
Number of Detectors per SCA per Band	Step 1
Focal Plane Model Parameters (Legendre Coefficients) (in radians)	Step 2
ACS-to-TIRS Alignment Matrix (3x3 orientation matrix) (see note #5)	Step 2
TIRS L1T Image (separated SCA)	Step 1
Image corner coordinates	Step 1
Pixel size (in meters)	Step 1
Image size	Step 1
Search image pixel data (10.8 micrometer band)	Step 1
DEM	Step 1
DEM corner coordinates	Step 1
Pixel size (in meters)	Step 1
DEM size	Step 1
Elevation data	Step 1
OLI Reference Image	Step 1
Image corner coordinates	Step 1
Pixel size (in meters)	Step 1
Image size	Step 1
Reference image pixel data (SWIR1 or SWIR2 band)	Step 1
Correlation Data File (output of Step 1)	Step 2
Correlation results in TIRS input space pixels	Step 2
LOS errors in radians	Step 2
Correlation results in output space pixels	Step 2

7.3.5.4 Outputs

Step 1: TIRS Alignment Setup
Correlation Data File (temporary output passed to Update step)
Correlation results in output space pixels
Correlation results in TIRS input space pixels
LOS errors in radians
Step 2: TIRS Alignment Update (see Table 1 below)
Report File (see Table 1 below for details)
Standard report header
Acquisition date
Ref (OLI)/Search (TIRS) image names
Constraint Type (Legendre or Angle)
Original TIRS-to-OLI roll-pitch-yaw
Estimated TIRS-to-OLI roll-pitch-yaw correction
Updated TIRS-to-OLI roll-pitch-yaw
Original TIRS-to-OLI alignment matrix (3x3)
Updated TIRS-to-OLI alignment matrix (3x3)
Number of SCAs
For each SCA:
SCA Number
Old Along- and Across-track Legendre coefficients (NSCAx2x4)
Along- and Across-track Legendre error (fit) coefficients (NSCAx2x4)
New Along- and Across-track Legendre coefficients (NSCAx2x4)
Pre-fit along- and across-track offset statistics (mean, stddev, RMSE)
Post-fit along- and across-track residual statistics (mean, stddev, RMSE)
Confidence level used for outlier rejection
Legendre polynomial fit order (see note #3)
Number of tie points used for current SCA
CPF LOS_LEGENDRE and ATTITUDE_PARAMETERS Groups (separate ODL-format files) (see note #6)
Effective Dates (embedded in output file names)
New ACS-to-TIRS alignment matrix (3x3)
New Legendre polynomial coefficients (NSCAx2x4)
Measure Tie Point Data
For each point:
SCA Number
Grid Cell Column Number
Nominal Output Space Line
Nominal Output Space Sample
Measured LOS Error Delta Line (in pixels)
Measured LOS Error Delta Sample (in pixels)
Measured LOS Error Along-Track Delta Angle (in microradians)
Measured LOS Error Across-Track Delta Angle (in microradians)
Tie Point State (outlier) Flag
Along-Track Fit Residual (in microradians)
Across-Track Fit Residual (in microradians)
TIRS Alignment Trending Database (see Table 1 below for details)
Geometric Characterization ID
Work Order ID
WRS path/row
Acquisition date
Ref (OLI) image name
Constraint Type (Legendre or Angle)

Original TIRS-to-OLI roll-pitch-yaw
Estimated TIRS-to-OLI roll-pitch-yaw correction
Updated TIRS-to-OLI roll-pitch-yaw
Number of SCAs
For each SCA:
SCA Number
Old Along- and Across-track Legendre coefficients (NSCAx2x4)
Along- and Across-track Legendre error (fit) coefficients (NSCAx2x4)
New Along- and Across-track Legendre coefficients (NSCAx2x4)
Pre-fit along- and across-track offset statistics (mean, stddev, RMSE)
Post-fit along- and across-track residual statistics (mean, stddev, RMSE)
Confidence level used for outlier rejection
Number of tie points used for current SCA

7.3.5.5 Options

TIRS Alignment Calibration Trending On/Off Switch

7.3.5.6 Prototype Code

Input to both executables is an ODL file containing all parameters needed by both programs; outputs are a binary tie point mensuration file (used internally only), an ASCII report file, two ASCII ODL-formatted CPF fragments, and trending data written to the stdout and captured in an ASCII log file.

The prototype code was compiled with the following options when creating the test data files:

-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The code units of the prototype implementation are briefly described here. Additional details are provided below for units that perform core algorithm processing logic.

TIRS_Align_Setup.c - This routine is the main driver for the setup portion of the TIRS alignment calibration.

get_tirs_align_parms.c - This function gets the input parameters from the ODL parameter file. It is used by both the setup and legendre executables.

check_images_match.c - This function checks to make sure the TIRS search, OLI reference, and DEM images all match. The corners of all the images should match to within half a pixel, and the reference and search images should be the same resolution (pixel size) and the same size. This function is an initial check to make sure that all the images are consistent before correlation is attempted. This function assumes the OLI reference image and the DEM are one-band images.

set_up_grid.c - This function reads the grid file into the grid data structure. The whole grid structure is returned so the caller can free all memory allocated when the grid was read using the grid deallocation call.

select_corr_pts.c - This function selects nominal correlation points evenly distributed about the center point of each grid cell (output space).

calc_input_space_errors.c - This function calculates the errors in input space pixels. This is done by first correlating in output space and converting the correlated locations to input space.

perform_correlation.c - This function performs the normalized gray-scale correlation at each point. It does this by invoking the correlation library routines described in the GCP Correlation Algorithm Description Document (e.g., math_submit_chip_to_corr).

map_coords_to_input_space.c - This function uses the inverse mapping coefficients in the grid to calculate the input space line/sample for each output space line/sample.

calc_los_errors.c - This function uses the tie point reference and search input space locations to calculate the angular line-of-sight errors.

output_correlation_info.c - This function writes the tie point correlation results to a file.

Generate_Legendre_Polynomials.c - This routine is the main driver for the Legendre polynomial and TIRS alignment angle estimation portion of TIRS alignment calibration.

read_correlation_info.c - This function reads the correlation information from the file generated by the output_correlation_info routine above.

filter_outliers.c - This function separates the correlation data into groups for each SCA for the X (sample) and Y (line) directions. It then finds the standard deviation for the points in each group. Outlier rejection is then performed on the points based on the tolerance selected by the user and the Student's T distribution. This procedure is described in the OLI Geometric Accuracy Assessment Algorithm Description Document.

calculate_point_weights.c - This function calculates the weight associated with each correlation point for doing the Legendre polynomial and TIRS alignment angle fit. Currently, this routine assigns the weight passed in to each point, effectively assigning each point an equal weight. Originally, it was thought that the correlation strength would factor into the weight, but that was determined to not be needed. This routine was left in to allow point-specific weight factors to be added at a later date.

fit_polynomials.c - This function performs the weighted least squares fit of the correlation data points (using the angular error) to find the TIRS alignment angle and Legendre error polynomials. Note that unlike the heritage OLI focal plane calibration procedure, TIRS alignment calibration performs a simultaneous solution for all parameters. This is necessary because the TIRS alignment angles are correlated with Legendre polynomial terms, linking the solutions for the three TIRS SCAs.

calculate_post_fit_residuals.c - This function calculates the residual statistics after the TIRS alignment angles and Legendre polynomial coefficients have been fit. This unit includes the calc_legendre_poly() function that calculates the Legendre polynomial for the input normalized detector value.

calculate_alignment_matrix.c – Computes the 3-by-3 alignment matrix corresponding to a set of 3 roll-pitch-yaw alignment angles.

create_tirs_alignment_report.c - This function generates a file reporting the results of the global TIRS alignment angle and SCA-specific Legendre polynomial fit calculations. The report file contents are

shown in Table1 below. This unit also includes the write_coeffs() function that writes an entire set of coefficients to the indicated output file.

trending_dummy.c - This function is a placeholder for the logic that will write the results of the TIRS alignment angle and SCA Legendre polynomial fit calculations to the geometric characterization database. In the prototype implementation the actual database output is replaced by dummy ASCII output to stdout.

update_alignment_parameters_cpf.c – Applies the computed TIRS-to-OLI alignment corrections to the current TIRS-to-OLI alignment matrix calculated from the CPF Attitude_To_OLI_Matrix and Attitude_To_TIRS_Matrix parameters, and then uses the updated TIRS-to-OLI alignment to compute an updated Attitude_To_TIRS_Matrix calibration parameter set.

write_alignment_matrix_ODL.c - This function writes the updated ATTITUDE_PARAMETERS parameter group of the CPF, in the ODL format used by the CPF, to an output file. This group contains the updated Attitude_To_TIRS_Matrix alignment parameters.

write_SCA_parameters_cpf.c - This function writes the updated LOS_LEGENDRE parameter group of the CPF, in the ODL format used by the CPF, to an output file.

7.3.5.7 Procedure

The TIRS Alignment Calibration Algorithm is used for on-orbit calibration of the TIRS-to-OLI instrument alignment as well as for the alignment of the lines-of-sight of the TIRS SCAs relative to each other. This calibration is necessary to meet the TIRS-to-OLI band registration, and the TIRS image registration, geodetic accuracy, and geometric accuracy requirements.

Procedure Overview
The TIRS alignment algorithm adjusts the overall TIRS field of view and each TIRS SCA to an OLI reference image. By simultaneously aligning the TIRS SCAs to a common reference, any measured inter-SCA misalignment is removed. Each TIRS SCA is correlated against a reference image created from an OLI SWIR band, acquired at approximately the same time. A new set of 3rd order Legendre LOS coefficients, representing updates or corrections to the original polynomials, are generated by fitting a set of coefficients to the measured LOS deviations. A set of roll-pitch-yaw angular adjustments are also computed to remove any alignment biases between the TIRS and OLI instruments.

Substantially cloud free scenes should be used for TIRS alignment calibration. The imagery should have ground control applied and terrain displacements removed, i.e. the imagery should be a terrain corrected (L1T) data set. Both the OLI SWIR and TIRS images should be path oriented and resampled to 30m output pixel size.

Stage 1: Setup – Correlate L1T Image with OLI Reference
An array of test points is generated for each TIRS SCA based upon the number of points per grid cell specified in the input parameters. The TIRS LOS projection grid is used to generate the test point array by spacing the test points at regular intervals in TIRS input space, and then computing the corresponding output space coordinates for each. Constructing the test point array in the TIRS input space ensures that the test points fall within the active area of each TIRS SCA.

Image windows extracted from the L1T image at the test point locations are correlated with corresponding windows extracted from the reference OLI SWIR image, using normalized gray scale correlation. This procedure is the same as that described in the OLI Focal Plane Calibration Algorithm Description Document. Since the expected offsets are small, the TIRS L1T and OLI SWIR image windows are the same size. The correlation procedure yields measured deviations (or correlation failure flag) in the line and sample directions, estimated to sub-pixel accuracy. These measured LOS deviations are in units of output space pixels.

The deviations measured in output space are converted to differences in LOS along- and across-track angles by mapping the reference point location from output space to TIRS input space and then mapping the search point location from output space to TIRS input space. The mappings are performed using the TIRS LOS projection grid that was used to resample the L1T image, and include the test point elevation interpolated from the input DEM. This three-dimensional output space to input space mapping (3d_ols2ils) is described in the TIRS Image Resampling Algorithm Description Document. This logic is identical to that used for OLI focal plane calibration. Once a TIRS input space location is found for both points, the LOS vectors are calculated for each input sample location using the TIRS LOS model. This is described in the Find LOS section of the TIRS LOS Projection Algorithm Description Document.

The angular LOS offsets in TIRS input space are then:

$$\text{along track offset} = \frac{r_x}{r_z} - \frac{s_x}{s_z} + (\text{input ref line} - \text{input search line}) * \text{along track IFOV} \quad (1\text{-}1)$$

$$\text{across track offset} = \frac{r_y}{r_z} - \frac{s_y}{s_z} \quad (1\text{-}2)$$

where:

r_x, r_y, r_z = reference x,y,z vector components of LOS

s_x, s_y, s_z = search x,y,z vector components of LOS

input reference line = input line location for reference point

input search line = input line location for search point

Stage 2: Update – Compute TIRS Alignment Calibration Update

A constrained least squares solution is used to generate the fit between the angular offsets and the corrections to the TIRS alignment angles and per-SCA Legendre polynomial coefficients. Constraints are necessary to separate the alignment angle estimates from the Legendre polynomial coefficients since the angular effects could be largely absorbed by the Legendre polynomials. The constraints are implemented as supplemental observations that enforce relationships between solution parameters.

There are two options for constraining the parameters. The first, default, option is to constrain the Legendre coefficients so that they do not attempt to model the rotation effects. The second option is to constrain the angular corrections to be zero. This effectively reduces the TIRS alignment calibration solution to be a simple focal plane calibration solution, similar to the OLI heritage. Both options use three additional constraint observations (described in more detail below) with an associated 3-by-3 constraint weight matrix. The weight matrix is a diagonal matrix with the diagonal terms containing a common weight, read from the CPF, for all 3 constraints.

Unlike the OLI focal plane calibration procedure which calibrates one SCA at a time, the TIRS alignment solution is simultaneous so that both the SCA-specific Legendre coefficients and the global alignment angles can be estimated. This also requires that both the along- and across-track Legendre coefficients be solved for at the same time. There are thus 27 unknowns to be solved for: 3 alignment angles + 3 SCAs * (4 along-track Legendre coefficients + 4 across-track Legendre coefficients).

There are N_k tie point observations in SCA k (k=1,2,3), so the total number of observations is $N = N_1 + N_2 + N_3$. The observation matrix is an Nx2 matrix containing the measured X and Y offsets in TIRS input space angular units. Note that, unlike the OLI focal plane calibration, there is a single observation matrix containing both the along track (X) offsets and the across track (Y) offsets.

$$\underset{2Nx1}{[B]} = \begin{bmatrix} x\,offset_1 \\ y\,offset_1 \\ x\,offset_2 \\ y\,offset_2 \\ \vdots \\ x\,offset_N \\ y\,offset_N \end{bmatrix} \qquad (2\text{-}1)$$

The design matrix is an 2Nx27 matrix with each row of the matrix containing the alignment angle partial derivatives and the Legendre polynomial terms associated with the reference sample location of the corresponding tie point measurement. The calculation of these angle partial derivatives and Legendre polynomial terms, as functions of the input sample location, is described below.

$$\underset{2Nx27}{[A]} = \begin{bmatrix} A_{0,1} & A_{1,1} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ A_{0,N1} & A_{1,N1} & 0 & 0 \\ A_{0,N1+1} & 0 & A_{1,N1+1} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ A_{0,N1+N2} & 0 & A_{1,N1+N2} & 0 \\ A_{0,N1+N2+1} & 0 & 0 & A_{1,N1+N2+1} \\ \vdots & \vdots & \vdots & \vdots \\ A_{0,N1+N2+N3} & 0 & 0 & A_{1,N1+N2+N3} \end{bmatrix} \qquad (2\text{-}2)$$

Where A_0 and A_1 are submatrices defined as:

$$A_{0,j} = \begin{bmatrix} 0 & -1 & y'_j \\ 1 & 0 & -x'_j \end{bmatrix}$$

$$A_{1,j} = \begin{bmatrix} l_{0,j} & l_{1,j} & l_{2,j} & l_{3,j} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & l_{0,j} & l_{1,j} & l_{2,j} & l_{3,j} \end{bmatrix}$$

And where:

x'_j = the x coordinate of tie point j, evaluated using the current estimate of the Legendre polynomials.

y'_j = the y coordinate of tie point j, evaluated using the current estimate of the Legendre polynomials.

$l_{i,j}$ = i^{th} Legendre polynomial term associated with the tie point location of the j^{th} measurement.

The Legendre polynomial terms (contained in the 2x8 $A_{1,j}$ submatrix) will be in the set of 8 columns associated with the SCA containing tie point j. Thus, columns 4-11 are associated with SCA 1, columns 12-19 are associated with SCA 2, and columns 20-27 are associated with SCA 3. In equation (2-2) tie points 1 through N_1 fall in SCA 1, points N_1+1 through N_1+N_2 fall in SCA 2, and points N_1+N_2+1 through N (=N_1+N_2+N_3) fall in SCA 3. Note that the partial derivative of the X offset with respect to the yaw correction is the tie point Y coordinate and the partial derivative of the Y offset with respect to the yaw correction is the –X tie point coordinate. Thus, the y'_j coordinate appears in the first (X observation) row of the A_0 submatrix and the $-x'_j$ coordinate appears in the second (Y observation) row of the A_0 submatrix.

The tie point observations are weighted by a diagonal weight matrix. The weight matrix [$\mathbf{W_t}$] is a 2Nx2N diagonal matrix where the diagonal elements are the tie point weight value read from the CPF.

$$\underset{2Nx2N}{[W_t]} = \begin{bmatrix} w_t & 0 & \cdots & 0 & 0 \\ 0 & w_t & 0 & \cdots & 0 \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ 0 & \cdots & \cdots & w_t & 0 \\ 0 & 0 & \cdots & 0 & w_t \end{bmatrix} \qquad (2\text{-}3)$$

where w_t = tie point weight

The weight matrix is included to make it possible to differentially weight the measured deviations based on correlation strength, but this is not implemented in the baseline algorithm. Instead, a common weight, read from the CPF, is used for all points. This weight value provides a relative weighting of the tie point observations relative to the constraints.

The form of the constraint observations depends upon the constraint option selected. For the default case, the Legendre coefficient adjustments are constrained to prevent them from modeling angular alignment effects. This is done by considering the effects of angular alignment changes at the center of each SCA. The x and y coordinates at the center of SCA k are:

$$x_0'_k = ax_{0k} - ax_{2k} / 2 \qquad (2\text{-}4)$$
$$y_0'_k = ay_{0k} - ay_{2k} / 2$$

where: ax_{0k} and ax_{2k} are the first and third x Legendre coefficients for SCA k, and ay_{0k} and ay_{2k} are the first and third y Legendre coefficients for SCA k.

So, the Legendre corrections at the center of each SCA are:

$$\Delta x_0'_k = \Delta ax_{0k} - \Delta ax_{2k} / 2 \qquad (2\text{-}5)$$
$$\Delta y_0'_k = \Delta ay_{0k} - \Delta ay_{2k} / 2$$

where: Δax_{0k} and Δax_{2k} are the first and third x Legendre corrections for SCA k,
Δay_{0k} and Δay_{2k} are the first and third y Legendre corrections for SCA k.

To control the modeling of alignment effects in the Legendre polynomials we constrain the Legendre adjustments to the locations of the SCA center points such that the mean center point x (pitch) and y (roll) locations do not change and such that the x (along-track) coordinates of the center points of the two outboard SCAs (1 and 3) do not change in opposite directions (yaw):

$$\begin{aligned} &\Delta ay_{01} + \Delta ay_{02} + \Delta ay_{03} - (\Delta ay_{21} + \Delta ay_{22} + \Delta ay_{23})/2 = 0 \quad \text{(roll)} \\ &\Delta ax_{01} + \Delta ax_{02} + \Delta ax_{03} - (\Delta ax_{21} + \Delta ax_{22} + \Delta ax_{23})/2 = 0 \quad \text{(pitch)} \\ &\Delta ax_{01} - \Delta ax_{21}/2 - \Delta ax_{03} + \Delta ax_{23} / 2 = 0 \quad \text{(yaw)} \end{aligned} \qquad (2\text{-}6)$$

The corresponding constraint matrix is:

$$\underset{3x27}{[C]} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(2-7)

The second constraint option fixes the alignment angle adjustments at zero, allowing all adjustments to be modeled through the Legendre coefficients. The corresponding constraint matrix is:

$$\underset{3x27}{[C]} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (2\text{-}8)$$

The constraint weight matrix $[W_c]$ is a 3-by-3 diagonal matrix containing the constraint weight read from the CPF:

$$\underset{3x3}{[W_c]} = \begin{bmatrix} w_c & 0 & 0 \\ 0 & w_c & 0 \\ 0 & 0 & w_c \end{bmatrix} \qquad (2\text{-}9)$$

where w_c = constraint weight

Note that the observation matrix is zero for the constraints no matter which constraint option is selected.

The solution for the updates to the TIRS alignment angles and to the Legendre LOS coefficients can be found from:

$$[\boldsymbol{\theta}] = \left([A]^T [W_t][A] + [C]^T [W_c][C]\right)^{-1} \left([A]^T [W_t][B]\right) \qquad (2\text{-}10)$$

The matrix [θ] is a 27x1 vector containing the corrections to be applied to the alignment angles and to the Legendre coefficients. These corrections are added to the original alignment angles and Legendre LOS coefficients to compute the updated TIRS alignment parameters.

$$[\boldsymbol{\theta}] = \begin{bmatrix} \Delta r \\ \Delta p \\ \Delta y \\ [\Delta ax_1] \\ [\Delta ay_1] \\ [\Delta ax_2] \\ [\Delta ay_2] \\ [\Delta ax_3] \\ [\Delta ay_3] \end{bmatrix} \quad [\Delta ax_k] = \begin{bmatrix} \Delta ax_{0k} \\ \Delta ax_{1k} \\ \Delta ax_{2k} \\ \Delta ax_{3k} \end{bmatrix} \qquad [\Delta ay_k] = \begin{bmatrix} \Delta ay_{0k} \\ \Delta ay_{1k} \\ \Delta ay_{2k} \\ \Delta ay_{3k} \end{bmatrix} \qquad (2\text{-}11)$$

The 10.8 micrometer band is used for TIRS alignment. TIRS band calibration uses the 10.8 micrometer band as the reference for the 12.0 micrometer band. A TIRS band alignment calibration should be performed following an update to the TIRS alignment calibration to avoid degrading the band-to-band registration.

Figure 1 shows the architecture for the setup portion of the TIRS Alignment Calibration algorithm.

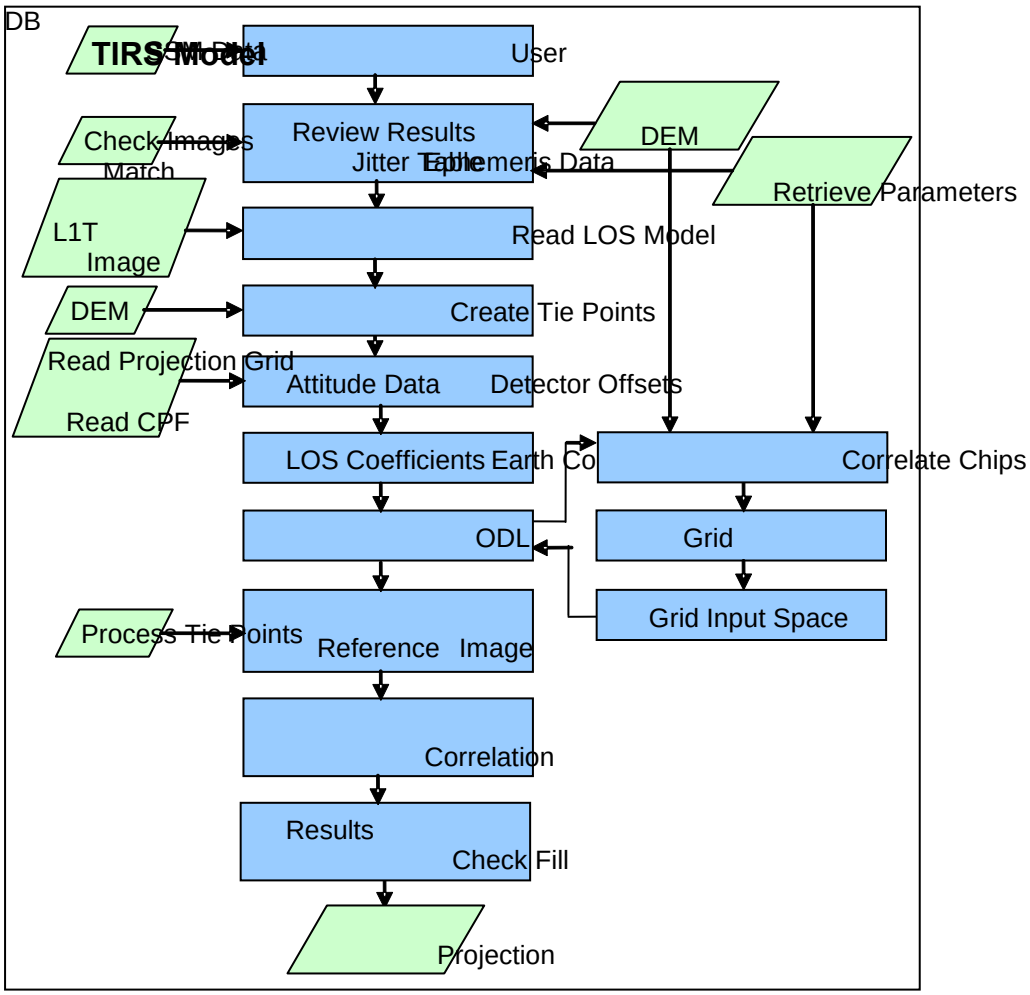


Figure 1: TIRS Alignment Calibration Setup Algorithm Architecture

Figure 2 shows the architecture of the alignment solution portion of the TIRS Alignment Calibration algorithm.

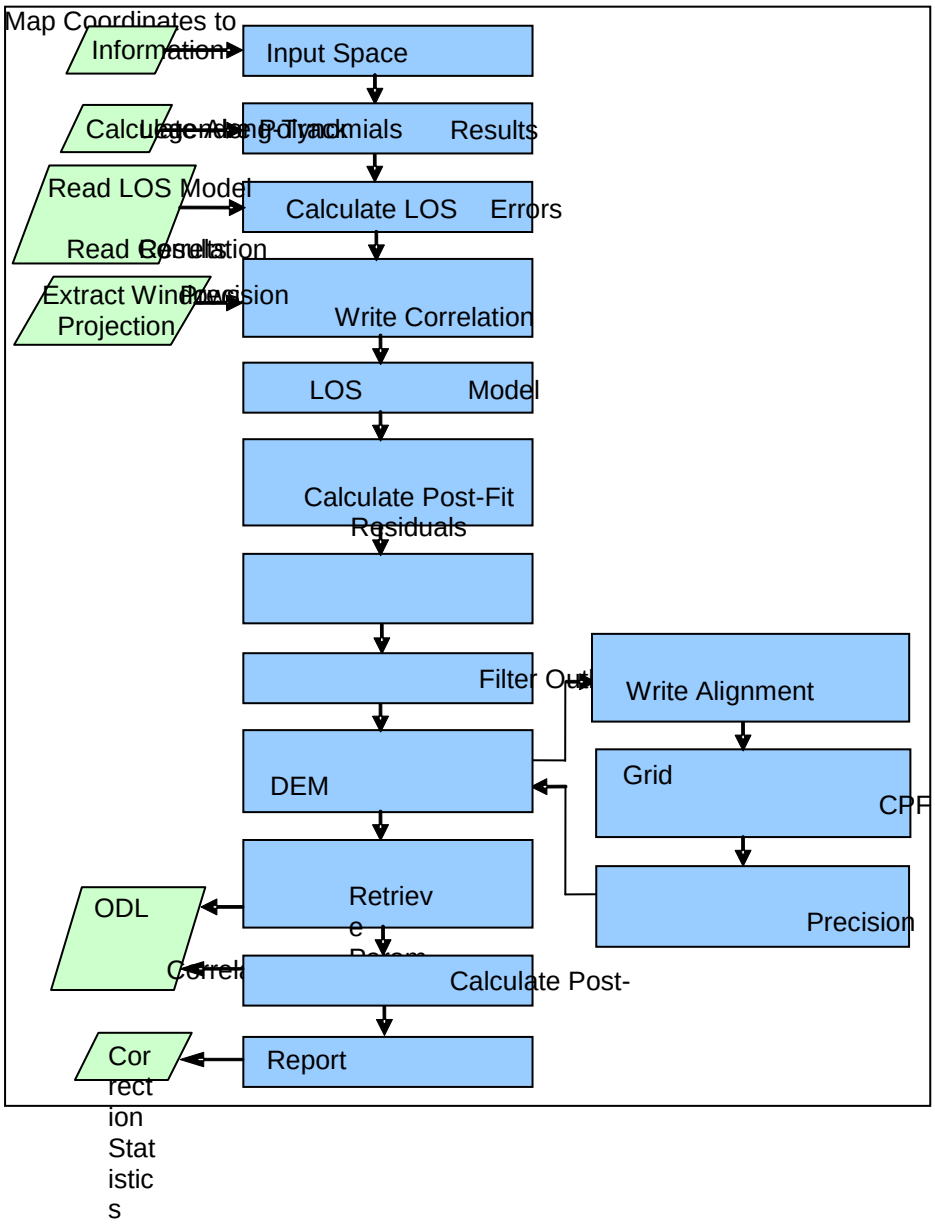


Figure 2: TIRS Alignment Calibration Update Algorithm Architecture

TIRS Alignment Cal Setup Sub-Algorithm (TIRS_Align_Setup)
This routine is the main driver for the setup portion of the TIRS alignment calibration. The setup portion consists of correlating points between the search TIRS image and the reference OLI SWIR image, and converting the correlation offsets into line-of-sight deviations that can be used to correct each SCA's detector array, modeled by a cubic Legendre polynomial. The results of this program are used in the second portion of TIRS alignment calibration, the generation of new TIRS-to-OLI alignment angles and TIRS SCA Legendre polynomials. This program creates a temporary output file that is read by the second portion of TIRS alignment calibration.

Get TIRS Alignment Parameters Sub-Algorithm (get_tirs_align_parms)
This function gets the input parameters from the input parameter files.

Check Images Match Sub-Algorithm (check_images_match)
This function checks to make sure the L1T TIRS search, OLI L1T SWIR reference, and DEM images all match. The corners of all the images should match to within half a pixel, and the reference and L1T search image should be the same resolution (pixel size) and the same size. This function is an initial check to make sure that all the images are consistent before correlation is attempted. This function assumes the OLI SWIR reference image and the DEM are one-band images.

Set Up Grid Sub-Algorithm (set_up_grid)
This function reads the TIRS grid file into the grid data structure. The whole grid structure is returned so the caller can free all memory allocated when the grid was read using the grid deallocation call.

Select Correlation Points Sub-Algorithm (select_corr_pts)
This function selects nominal correlation points evenly distributed about the center point of each grid cell (output space). To ensure evenly distributed tie point locations during correlation, locations are defined to lie at the center of each resampling grid cell, or sub-cell. There will be pts_per_cell equally-spaced points per grid cell. For example, if there are 4 points per cell, they will be placed as shown in Figure 3.

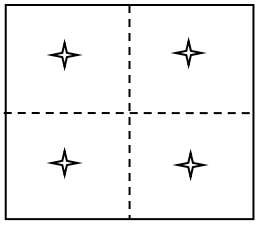


Figure 3: Correlation Point Placement in Grid Cell

The cell is divided into a 2-by-2 grid of 4 sub-cells, and each sub-cell is divided in half to place the point in the middle, yielding points at (0.25,0.25), (0.75,0.25), (0.25,0.75), and (0.75,0.75).

The calculation of the output space line/sample coordinates of the tie points is done as follows:

a) Compute number of rows and columns of tie points in each cell.
   ncol = (int)ceiling(sqrt(pts_per_cell))
   nrow = (int)ceiling((double)pts_per_cell/(double)ncol)
   This creates an array of tie points containing at least pts_per_cell points.

b) For each tie point, i = 1 to ncol and j = 1 to nrow:
   b1) Compute the grid cell fractional location (cfrac,rfrac).

$$cfrac = \frac{2i-1}{2\ ncol} \qquad rfrac = \frac{2j-1}{2\ nrow}$$

   b2) Compute the output space line, ol_{ij}, using bilinear interpolation on the output line numbers at the grid cell corners, where l_{UL}, l_{UR}, l_{LL}, and l_{LR} are the output space line coordinates at the grid cell upper-left, upper-right, lower-left, and lower-right corners, respectively:

$$ol_{ij} = l_{UL} * (1-cfrac)*(1-rfrac) + l_{UR} * cfrac * (1-rfrac) + l_{LL} * (1-cfrac) * rfrac$$

+ l_{LR} * cfrac * rfrac

b3) Compute the output space sample, os_{ij}, using bilinear interpolation on the output sample numbers at the grid cell corners, where s_{UL}, s_{UR}, s_{LL}, and s_{LR} are the output space sample coordinates at the grid cell upper-left, upper-right, lower-left, and lower-right corners, respectively:

os_{ij} = s_{UL} * (1-cfrac)*(1-rfrac)
+ s_{UR} * cfrac * (1-rfrac)
+ s_{LL} * (1-cfrac) * rfrac
+ s_{LR} * cfrac * rfrac

Note that the bilinear weights are the same for the line and sample computations and only need be computed once.

The heritage version of this sub-algorithm locates the points by computing the intersection of the cell diagonals and then calculating offsets from that point. It is more straightforward to simply use bilinear interpolation, as described above, to calculate the tie point output space coordinates, so this unit will be reworked from the heritage implementation.

Calculate Input Space Errors Sub-Algorithm (calc_input_space_errors)
This function calculates the errors in TIRS input space pixels. This is done by first correlating in output space and converting the correlated locations to input space.

Perform Correlation Sub-Algorithm (perform_correlation)
This function performs the normalized gray-scale correlation at each point. It does this by invoking the correlation library routines described in the GCP Correlation Algorithm Description Document.

Map Coordinates to Input Space Sub-Algorithm (map_coords_to_input_space)
This function uses the inverse mapping coefficients in the grid to calculate the TIRS input space line/sample for each output space line/sample. It does this for both the reference (OLI SWIR) image line/sample location and the search (TIRS L1T) image line/sample location, mapping both to TIRS input space.

a) For each SCA

a1) For each tie point

a1.1) Interpolate a height from the DEM at the location corresponding to the tie point reference image line/sample coordinates (ALIAS xxx_get_elevation).

a1.2) Map the reference output line/sample location to its corresponding input line/sample location using axx_3d_ols2ils routine

a1.3) Interpolate a height from the DEM at the location corresponding to the tie point search image line/sample coordinates (ALIAS xxx_get_elevation).

a1.4) Map the search output line/sample location to its corresponding input line/sample location using the axx_3d_ols2ils

Calculate LOS Errors Sub-Algorithm (calc_los_errors)

This function uses the tie point reference and search TIRS input space locations to calculate the angular line-of-sight errors.

b) For each SCA

b1) For each tie point

b1.1) Calculate reference line of sight vector for sample location using the nominal detector type and the precision LOS model (ALIAS axx_findlos).

b1.2) Calculate the search LOS vector for sample location using the nominal detector type and the precision LOS model.

b1.3) Calculate the deviations in terms of the difference in the LOS along and across track angles:

along-track LOS error = ref los.x/los.z – srch los.x/los.z
+ input line error * along-track IFOV

across-track LOS error = ref los.y/los.z – srch los.y/los.z

Output Correlation Information Sub-Algorithm (output_correlation_info)

This function writes the tie point correlation results to a file. The correlation points are dumped to a binary file, so the second phase of TIRS alignment calibration (TIRS alignment update) can read them directly back in. First, a long integer is written to indicate the number of records, and then all the records are written. Each record contains the following fields:

Type	Field	Description
int	sca_number	SCA number (0-relative)
int	grid_column	grid column number (0-relative)
int	grid_row	grid row number (0-relative)
double	nom_os_pt.line	nominal output space point line
double	nom_os_pt.samp	nominal output space point sample
double	ref_os_pt.line	reference output space line
double	ref_os_pt.samp	reference output space sample
double	srch_os_pt.line	search output space line
double	srch_os_pt.samp	search output space sample
double	ref_is_pt.line	reference TIRS input space line
double	ref_is_pt.samp	reference TIRS input space sample
double	srch_is_pt.line	search TIRS input space line
double	srch_is_pt.samp	search TIRS input space sample
double	los_err.line	angular along-track LOS error
double	los_err.samp	angular across-track LOS error
double	los_err_pix.line	line LOS error in pixels
double	los_err_pix.samp	sample LOS error in pixels
double	correlation_accuracy	correlation accuracy
ActiveFlag	active_flag	correlation success flag
double	pt_weight	point weight for use in fit
double	fit_residual.line	line residual from fit of pts
double	fit_residual.samp	sample residual from fit of pts

This is not a human-readable (ASCII) file, because it is only used to transport information from the first phase of calibration to the second. If the file already exists, it will be overwritten.

TIRS Alignment Update Sub-Algorithm (Generate_Legendre_Polynomials)

This routine is the main driver for the least squares solution and alignment update portion of TIRS alignment calibration. The TIRS alignment update portion reads the results of the TIRS alignment setup, filters the outliers, fits the data to a set of angular alignment corrections and Legendre polynomial corrections, updates the TIRS-to-OLI alignment and TIRS SCA models, and generates output reports. This process is outlined below.

a) Initialize the normal equation matrix [N] (27x27) and constant vector [L] (27x1) to zero.

b) Process each tie point (j)

b.1) Build design matrix [A_j].

Calculate normalized detector for reference sample location. Note that in this context the "detector" number is the input sample number within the SCA containing the tie point (SCA #k).

$$\text{normalized detector} = \frac{2 * \text{detector}}{(\text{number of detectors} - 1)} - 1$$

where:
detector = reference sample location (0 ... Ndet-1)
number of detectors = number of detectors in current SCA

Calculate two rows of design matrix associated with current tie point:

$l_{0,j} = 1$
$l_{1,j}$ = normalized detector
$l_{2,j}$ = (3*(normalized detector)^2 – 1) / 2
$l_{3,j}$ = normalized detector *(5 *(normalized detector)^2 – 3) / 2
where j = tie point number

$$x'_j = \sum_{i=0}^{3} ax_{ik}\, l_{i,j}$$

$$y'_j = \sum_{i=0}^{3} ay_{ik}\, l_{i,j}$$

where: j = tie point number
k = SCA number

$$\underset{2x27}{[A_j]} = [[A_0] \quad [A_1] \quad [A_2] \quad [A_3]]$$

where:

$$\underset{2x3}{[A_0]} = \begin{bmatrix} 0 & -1 & y'_j \\ 1 & 0 & -x'_j \end{bmatrix}$$

$$\underset{2x8}{[A_i]} = \begin{bmatrix} l_{0,j} & l_{1,j} & l_{2,j} & l_{3,j} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & l_{0,j} & l_{1,j} & l_{2,j} & l_{3,j} \end{bmatrix} \quad \text{if k = i}$$

$$\underset{2x8}{[A_i]} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{otherwise}$$

b.2) Build weight matrix [W_j] using the (fixed) input weight value.

$$\underset{2x2}{[W_j]} = \begin{bmatrix} w_t & 0 \\ 0 & w_t \end{bmatrix}$$

b.3) Build the observation matrix [B_j] from the measured x and y offsets in terms of angular differences.

$$\underset{2x1}{[B_j]} = \begin{bmatrix} x\,offset_j \\ y\,offset_j \end{bmatrix}$$

b.4) Add this observation to the normal equations matrix [N] and to the constant vector [L].

$$[N] = [N] + [A_j]^T [W_j] [A_j]$$
$$[L] = [L] + [A_j]^T [W_j] [B_j]$$

c) Add the constraints.

c.1) Form the constraint design matrix [C] based upon the user-selected constraint option, using equation (2-7) or equation (2-8) above.

c.2) Form the constraint weight matrix [W_c] per equation (2-9) above.

c.3) Add the constraint contribution to the normal equations matrix [N]. Note that there is no constraint contribution to the constant vector [L].

$$[N] = [N] + [C]^T [W_c] [C]$$

d) Solve for alignment angle and Legendre coefficient corrections using weighted least squares routine (see the Fit Parameters sub-algorithm below).

e) Calculate pre-fit statistics from the original measured deviations.

e.1) Calculate statistics for along- and across-track offsets used in b.3.

Compute mean, standard deviation and RMSE for the offsets, grouping by SCA. Values are calculated for along- and across-track directions independently. For each SCA k=1,2,3:

For along-track offsets in SCA k:
e.1.1) Calculate x mean
e.1.2) Calculate x standard deviation
e.1.3) Calculate x RMSE

For across-track offsets in SCA k:

e.1.4) Calculate y mean
e.1.5) Calculate y standard deviation
e.1.6) Calculate y RMSE

f) Calculate post fit residual statistics for correction coefficients

Post-fit residuals are calculated by updating the original measured offsets/deviations used in step b.3 above using the alignment angle and Legendre polynomial corrections. The differences between the original measurements and the offsets modeled by the correction parameters are the residuals. The post-fit statistics are calculated on these residuals.

f.1) For each tie point

f.1.1) Calculate normalized detector for reference sample location (as shown in b.1) and construct the design matrix as shown in b.1.

f.1.2) Calculate the modeled LOS angle corrections by multiplying the design matrix by the [θ] solution vector.

f.1.3) Find the residuals as the difference between the original angular offsets and the LOS angle corrections from f.1.2.

f.2) Calculate statistics for the along- and across-track residuals calculated in f.1.

Compute mean, standard deviation and RMSE for residuals, grouping by SCA. Values are calculated for along- and across-track directions independently. For each SCA k=1,2,3:

For along-track residuals:
f.2.1) Calculate x mean
f.2.2) Calculate x standard deviation
f.2.3) Calculate x RMSE

For across-track residuals:
f.2.4) Calculate y mean
f.2.5) Calculate y standard deviation
f.2.6) Calculate y RMSE

g) For each SCA, add the correction coefficients to original Legendre LOS coefficients (see note #4):

$$\text{new along legendre}_{i,sca} = \text{update along legendre}_{i,sca} + \text{old along legendre}_{i,sca}$$

$$\text{new across legendre}_{i,sca} = \text{update across legendre}_{i,sca} + \text{old across legendre}_{i,sca}$$

where:
i = 0,1,2,3 Legendre polynomial number
sca = SCA number

h) Use the computed roll, pitch, and yaw alignment corrections to update the TIRS-to-OLI rotation matrix.

h.1) Compute the delta rotation matrix [ΔM] from the Δr, Δp, and Δy corrections.

$$[\Delta M] = \begin{bmatrix} \cos(\Delta p)\cos(\Delta y) & \sin(\Delta r)\sin(\Delta p)\cos(\Delta y) + \cos(\Delta r)\sin(\Delta y) & \sin(\Delta r)\sin(\Delta y) - \cos(\Delta r)\sin(\Delta p)\cos(\Delta y) \\ -\cos(\Delta p)\sin(\Delta y) & \cos(\Delta r)\cos(\Delta y) - \sin(\Delta r)\sin(\Delta p)\sin(\Delta y) & \cos(\Delta r)\sin(\Delta p)\sin(\Delta y) + \sin(\Delta r)\cos(\Delta y) \\ \sin(\Delta p) & -\sin(\Delta r)\cos(\Delta p) & \cos(\Delta r)\cos(\Delta p) \end{bmatrix}$$

h.2) Combine the delta rotation matrix [ΔM] with the original rotation matrix [TIRS2OLI], derived from the ACS-to-OLI and ACS-to-TIRS matrices in the CPF, to form the updated rotation matrix [TIRS2OLI]'.

$$[\text{TIRS2OLI}] = [\text{ACS2OLI}]\,[\text{ACS2TIRS}]^T$$

$$[\text{TIRS2OLI}]' = [\text{TIRS2OLI}]\,[\Delta M]$$

h.3) Compute original and updated TIRS-to-OLI alignment angles.

$$\text{Roll} = \text{atan}(-[\text{TIRS2OLI}]_{3,2} / [\text{TIRS2OLI}]_{3,3})$$
$$\text{Pitch} = \text{asin}([\text{TIRS2OLI}]_{3,1})$$
$$\text{Yaw} = \text{atan}(-[\text{TIRS2OLI}]_{2,1} / [\text{TIRS2OLI}]_{1,1})$$

$$\text{Roll}' = \text{atan}(-[\text{TIRS2OLI}]'_{3,2} / [\text{TIRS2OLI}]'_{3,3})$$
$$\text{Pitch}' = \text{asin}([\text{TIRS2OLI}]'_{3,1})$$
$$\text{Yaw}' = \text{atan}(-[\text{TIRS2OLI}]'_{2,1} / [\text{TIRS2OLI}]'_{1,1})$$

h.4) Compute the updated ACS-to-TIRS rotation matrix.

Compute the updated OLI-to-TIRS rotation matrix as the transpose of the updated TIRS-to-OLI matrix:

$$[\text{OLI2TIRS}]' = [\text{TIRS2OLI}]'^T$$

Compute the updated ACS-to-TIRS rotation matrix using the ACS-to-OLI matrix from the CPF and the updated OLI-to-TIRS rotation matrix:

$$[\text{ACS2TIRS}]' = [\text{OLI2TIRS}]'\,[\text{ACS2OLI}]$$

Read Correlation Information Sub-Algorithm (read_correlation_info)
This function reads the correlation information from the file generated by the Output Correlation Information sub-algorithm above.

Filter Outliers Sub-Algorithm (filter_outliers)
This function separates the focal plane correlation data into groups for each SCA for the X (sample) and Y (line) directions. It then finds the standard deviation for the points in each group. Outlier rejection is then performed on the points based on the tolerance selected by the user and the Student's T distribution. This procedure is described in the Geometric Accuracy Assessment Algorithm Description Document.

Calculate Point Weights Sub-Algorithm (calculate_point_weights)

This function calculates the weight associated with each correlation point for doing the Legendre polynomial fit.

Currently, this routine assigns the weight passed in to each point, effectively assigning each point an equal weight. Originally, it was thought that the correlation strength would factor into the weight, but that was determined to not be needed. This routine was left in to allow point-specific weight factors to be added at a later date.

Fit Parameters Sub-Algorithm (fit_polynomials)
This function performs the weighted least squares fit of the correlation data points (using the angular error) to find the alignment angle corrections and the Legendre error polynomials. The least squares correction parameter vector $[\theta]$ is given by solving:

$$[N]\ [\theta] = [L]$$

Where:
[N] is the [27 x 27] normal equations matrix

$[\theta]$ is the [27 x 1] unknown vector containing the alignment angle and Legendre coefficient corrections we are looking for

[L] is the [27 x 1] constant vector

Solving the above equation for $[\theta]$ yields: $[\theta] = ([N])^{-1} * [L]$

Calculate Post-fit Residuals Sub-Algorithm (calculate_post_fit_residuals)
This function calculates the residual statistics, as described in step f) above, after the alignment corrections and Legendre polynomial coefficient corrections have been calculated.

Calculate Legendre Polynomial Sub-Algorithm (calc_legendre_poly)
This function calculates the Legendre polynomial for the input normalized detector value, x:

$along = coeff\_along_0 + coeff\_along_1\, x + coeff\_along_2\, (3*x^2 - 1)/2 + coeff\_along_3\, x*(5*x^2 - 3)/2$

$across = coeff\_across_0 + coeff\_across_1\, x + coeff\_across_2\, (3*x^2 - 1)/2 + coeff\_across_3\, x*(5*x^2 - 3)/2$

Create TIRS Alignment Report Sub-Algorithm (create_tirs_alignment_report)
This function generates a file reporting the results of the TIRS-to-OLI alignment angle and TIRS SCA Legendre polynomial fit calculations. The report file contents are shown in Table 1 below.

Write Coefficients Sub-Algorithm (write_coeffs)
This function writes an entire set of coefficients to the indicated output file.

Write TIRS Alignment Calibration Results to Characterization Database (trend_to_database)
This function writes the results of the TIRS-to-OLI alignment angle and TIRS SCA Legendre polynomial fit calculations to the geometric characterization database. The output is only written to the database if the post-fit along- and across-track RMSE statistics are all below the threshold

values specified in the CPF (the trending metrics). The characterization database output is listed in Table 1 below.

Write SCA Parameters CPF Sub-Algorithm (write_SCA_parameters_cpf)
This function writes the updated Legendre coefficients to a new LOS_LEGENDRE CPF parameter group, in the ODL format used by the CPF, to a separate ASCII output file. All the CPF LOS_LEGENDRE parameter values except for the new Legendre coefficients are extracted from the original CPF or the LOS Model structure. Current plans call for actual calibration updates to be based on multiple scene results extracted from the characterization database, so this capability is primarily a convenience for testing purposes.

Write TIRS-to-OLI Alignment Parameters CPF Sub-Algorithm (update_alignment_parameters_cpf)
This function writes the ATTITUDE_PARAMETERS parameter group of the CPF, in the ODL format used by the CPF, to a separate ASCII output file. The updated ACS-to-TIRS rotation matrix computed in h.4) above is written to the parameter group along with the current values for all other parameters in that group. Current plans call for actual calibration updates to be based on multiple scene results extracted from the characterization database, so this capability is primarily a convenience for testing purposes.

Algorithm Output Details
The contents of the output TIRS alignment calibration report file and the corresponding geometric characterization database outputs are summarized in Table 1 below. All fields are written to the output report file but only those with "Yes" in the "Database Output" column are written to the characterization database. Note that the first eleven fields listed constitute the standard report header.

Field	Description	Database Output
585. Date and time	586. Date (day of week, month, day of month, year) and time of file creation.	587. Yes
588. Spacecraft and instrument source	589. LDCM and TIRS	590. Yes
591. Processing Center	592. EROS Data Center SVT	593. Yes
594. Work order ID	595. Work order ID associated with processing (blank if not applicable)	596. Yes
597. WRS path	598. WRS path number	599. Yes
600. WRS row	601. WRS row number	602. Yes
603. Software version	604. Software version used to create report	605. Yes
606. Off-nadir angle	607. Scene off-nadir roll angle (in degrees) (only nadir-viewing scenes are used for TIRS alignment)	608. Yes

609. Acquisition type	610. Earth, Lunar, or Stellar (only Earth-viewing scenes are used for TIRS alignment calibration)	611. Yes
612. Geo Char ID	613. Geometric Characterization ID	614. Yes
615. L1T image file	616. Name of TIRS L1T used to measure tie points	617. No
Acquisition date	Date of L1T image acquisition (new)	Yes
Reference image file	Name of reference (OLI) image used to measure tie points	Yes
Original TIRS-to-OLI angles	Original TIRS-to-OLI roll-pitch-yaw alignment angles in radians (new)	Yes
TIRS-to-OLI correction angles	Estimated roll-pitch-yaw corrections to the TIRS-to-OLI alignment knowledge in radians (new)	Yes
Update TIRS-to-OLI angles	Updated TIRS-to-OLI roll-pitch-yaw alignment angles in radians (new)	Yes
Original TIRS alignment matrix	Original 3x3 TIRS-to-OLI alignment matrix (new)	No
Updated TIRS alignment matrix	Updated 3x3 TIRS-to-OLI alignment matrix (new)	No
Confidence Level	Confidence level used for outlier rejection	Yes
Fit Order	Order of Legendre fit	Yes
Number of SCAs	Number of SCAs calibrated (3)	Yes
For each SCA:		
SCA Number	Number of the current SCA (1-3)	Yes
Original AT Legendre coeffs	Original along-track Legendre coefficients: a0, a1, a2, a3	Yes
Original XT Legendre coeffs	Original across-track Legendre coefficients: b0, b1, b2, b3	Yes
Error AT Legendre coeffs.	The computed updates to the along-track Legendre coefficients: c0, c1, c2, c3	Yes
Error XT Legendre coeffs.	The computed updates to the across-track Legendre coefficients: d0, d1, d2, d3	Yes
New AT Legendre coeffs	New along-track Legendre coefficients: a'0, a'1, a'2, a'3	Yes
New XT Legendre coeffs	New across-track Legendre coefficients: b'0, b'1, b'2, b'3	Yes
Pre-fit AT statistics	Pre-fit along-track offset mean, standard deviation, and RMSE statistics	Yes
Pre-fit XT statistics	Pre-fit across-track offset mean, standard deviation, and RMSE statistics	Yes
Post-fit AT residual statistics	Post-fit along-track residual mean, standard deviation, and RMSE statistics	Yes
Post-fit XT residual statistics	Post-fit across-track residual mean, standard deviation, and RMSE statistics	Yes
Number of Points	Number of tie points used for current SCA	Yes
CPF Group:	Written to a separate output ODL file	
Effective Date Begin	Beginning effective date of CPF group (from the original CPF): YYYY-MM-DD	No

Effective Data End	Ending effective date of CPF group (from the original CPF): YYYY-MM-DD	No
ACS-to-TIRS rotation matrix	Updated 3x3 attitude control system-to-TIRS rotation matrix	No
Number of SCAs	Number of SCAs (3): Num_SCA = 3	No
For each SCA:		
New Legendre polynomial coefficients	Four (one per band/row) arrays of four along-track Legendre coefficients followed by four arrays of four across-track Legendre coefficients.	No
Tie Point Data:	For each tie point:	
SCA Number	SCA where the tie point was measured	No
Grid Cell Column Number	Column number of the grid cell containing the tie point	No
Nominal Output Space Line	Predicted tie point output space line location	No
Nominal Output Space Sample	Predicted tie point output space sample location	No
LOS Line Error	Measured LOS error delta line (in pixels)	No
LOS Sample Error	Measured LOS error delta sample (in pixels)	No
LOS AT Error	Measured LOS error along-track delta angle (in microradians)	No
LOS XT Error	Measured LOS error across-track delta angle (in microradians)	No
State Flag	Tie point state (outlier) flag	No
AT Fit Residual	Along-track fit residual (in microradians)	No
XT Fit Residual	Across-track fit residual (in microradians)	No

Table 1: TIRS Alignment Calibration Output Details

Accessing the TIRS Alignment Results in the Characterization Database

Though not part of the formal TIRS alignment calibration algorithm, some comments regarding the anticipated methods of accessing and analyzing the individual scene TIRS alignment calibration results stored in the characterization database may assist with the design of the characterization database.

The database output from the TIRS alignment calibration algorithm will be accessed by a data extraction tool that queries the characterization database to retrieve TIRS alignment calibration results from multiple scenes. The only processing required on the returned results is to compute the average "new" TIRS-to-OLI alignment angles and the average "new" Legendre coefficients for each SCA across all returned scenes. The returned scene results and computed mean alignment angles and mean Legendre coefficient values will be output in a report containing a comma-delimited table of the retrieved trending results as well as the summary averages.

The geometric results would typically be queried by acquisition date and/or WRS path/row. The most common query would be based on acquisition date range, for example, selecting all of the results for a given calendar quarter:

Acquisition_Date is between 01APR2012 and 30JUN2012

The average alignment angles would be calculated from the updated alignment angles for the individual scenes returned, as:

$$Angle_{j,net} = \frac{1}{numScene} \sum_{i=1}^{numScene} Angle_{j,i}$$

for angle j = roll, pitch, yaw.

The average coefficients would be calculated from the "new" Legendre coefficients for the individual scenes returned, as:

$$Coeff_{SCA,j,net} = \frac{1}{numScene} \sum_{i=1}^{numScene} Coeff_{SCA,j,i}$$

for coefficient j (j=0,1,2,3) for each SCA (SCA=1,2,3).

The query results would be formatted in a set of comma-delimited records (for ease of ingest into Microsoft Excel), one record per scene. Each record would contain all of the "header" fields written to the characterization database (items with "Yes" in the rightmost column of Table 1 above) but only the "new" alignment angles and the "new" Legendre coefficients for each SCA. The other fields would be retrieved using general purpose database access tools, if and when desired. A header row containing the field names should precede the database records.

Following the scene records the average alignment angles and Legendre coefficients should be written out in the same CPF/ODL syntax used in the report file. This output uses the same structure shown in the final row in Table 1 above, but contains the average, rather than a single scene's, angles and Legendre coefficients.

7.3.5.8 Maturity

Most of the OLI focal plane alignment calibration logic was reusable, but the TIRS version was adapted to include the TIRS-to-OLI alignment computation. The focal plane alignment calibration logic were also adapted to the TIRS sensor parameters:

3. There are 3 separate SCAs to calibrate (vs. 14 OLI SCAs).
4. Analysis of the TIRS optical model indicated that the heritage Legendre polynomial order of 2 is not sufficiently accurate for the TIRS due to the significantly longer SCAs. Prototype tests indicate that a Legendre polynomial order of 3 is sufficient. This was a relatively minor change to the algorithm.
5. Since the alignment angle and Legendre coefficient updates are being computed together, the solution must be simultaneous. This is a departure from the heritage method in which each SCA was calibrated separately.

7.3.5.9 Notes

Some additional background assumptions and notes include:

6. The 10.8 micrometer thermal band will be used to provide the geometric reference for the TIRS instrument. If subsequent band correlation studies show that the 12.0 micrometer band provides superior correlation performance relative to the OLI SWIR bands, it could be used as the reference instead and this decision will be revisited. Such a change would affect the TIRS band alignment calibration algorithm as well.

7. The input TIRS and OLI L1T images are treated as separate inputs in the baseline TIRS algorithm. These could ultimately be contained in the same output image, but since the TIRS image is SCA-separated and the OLI image is SCA-combined it may be best to keep them as two distinct input images even if merged OLI and TIRS images can be produced.
8. The TIRS focal plane model uses third order Legendre coefficients to model the line-of-sight directions for each SCA, as noted in maturity note #2 above.
9. The baseline assumption is that Legendre coefficient sets will be stored in the CPF for both active detector rows for each band (10.8 and 12.0 micrometer) on each SCA, but that only the primary detector set will be calibrated. Only the coefficients for the primary row for the 10.8 micrometer band will be updated by this calibration procedure. This assumption may be modified if it is decided that the TIRS CPF will contain only Legendre coefficients for the primary detector rows, in which case there will only be two sets of Legendre coefficients in the CPF, or if the Legendre coefficients for the redundant rows are to be maintained by the calibration procedures, in which case the computed updates will be added to the Legendre coefficients from both rows for the 10.8 micrometer band.
10. The ATTITUDE_PARAMETERS CPF parameter group contains the ACS-to-TIRS and ACS-to-OLI alignment matrices. These two matrices are related by the TIRS-to-OLI alignment matrix, which is maintained by the TIRS alignment calibration algorithm, as follows: [ACS2OLI] = [TIRS2OLI] [ACS2TIRS]. To avoid the redundancy inherent in retaining all three of these matrices in the CPF, the [TIRS2OLI] matrix is constructed, when needed, from the ACS-to-sensor matrices as:

    $$[TIRS2OLI] = [ACS2OLI]\,[ACS2TIRS]^{-1}$$

    This [TIRS2OLI] matrix is updated by the TIRS alignment calibration procedure and the result is then used to update the [ACS2TIRS] matrix in the CPF as:

    $$[ACS2TIRS] = [TIRS2OLI]^{-1}\,[ACS2OLI].$$
11. The LOS_LEGENDRE CPF group contains the TIRS focal plane model in the form of the along-track and across-track Legendre coefficients updated by this algorithm.

7.3.6 TIRS Band Registration Accuracy Assessment

7.3.6.1 Background/Introduction

The TIRS Band Registration Accuracy Assessment Algorithm (BRAA), or the Band-to-Band (B2B) Characterization process, measures the relative band alignment between the spectral bands on each Sensor Chip Assembly (SCA) for the TIRS instrument. The displacement for every pair-wise combination of all bands requested for assessment is measured on each SCA, creating a set of band-to-band measurements for each SCA. The number of bands available for assessment could be from two (the primary 10.8 and 12 micrometer thermal bands) to four for TIRS only or as many as thirteen - 9 OLI reflective bands and two separate representations of the two TIRS spectral bands derived from the primary and redundant TIRS detector rows - depending on the contents of the input L1T image(s). The residuals measured from the B2B characterization process will be used to assess the accuracy of the band-to-band registration of the TIRS instrument, and if need be, used as input to the band calibration algorithm in order to calculate new line-of-sight (LOS) parameters for the Calibration Parameter File (CPF).

The B2B characterization process works by choosing tie point locations within band pairs of each SCA, extracting windows of imagery from each band and performing grey scale correlation on the image windows. Several criteria are used in determining whether the correlation process was successful. These criteria include measured displacement and strength of the correlation peak. The sub-pixel location of the measured offset is calculated by fitting a 2^{nd} order polynomial around the discrete correlation surface and solving for the fractional peak location of the fitted polynomial. The total offset measured is then the integer location of the correlation peak plus the sub-pixel location calculated.

There are several options available for processing data through the Band Registration Accuracy Assessment algorithm. These include choosing evenly spaced points for location of the windows extracted, choosing to use the TIRS LOS projection grid for determining window locations in order to avoid fill within the image files, specifying the bands to process, and specifying the valid pixel range to use during correlation. Note that, unlike OLI, in which individual SCAs can be selected for processing (to support the analysis of lunar data), we always process all three TIRS SCAs.

The TIRS B2B characterization algorithm will typically be exercised in one of two different modes, depending upon the input image provided. When operating on SCA-separated TIRS images, tie points based on the TIRS LOS projection grid would be generated to evaluate the internal registration of the TIRS spectral bands. The resulting tie point measurements would be suitable for subsequent use in TIRS band alignment calibration. When operating on SCA-combined images containing both TIRS and OLI data, a regular array of tie points would be used to measure the registration of all selected TIRS and OLI band pair combinations. These tie point results would be suitable for characterizing TIRS to OLI band registration performance.

The TIRS B2B characterization and OLI B2B characterization algorithms are very similar and could potentially be combined at some point in the future. This revision of the TIRS B2B algorithm addresses only the TIRS-only and OLI/TIRS combined cases of band registration accuracy assessment.

7.3.6.2 Dependencies

The TIRS BRAA assumes that a cloud free Earth viewing L1T image has been generated and depending on the tie point selection type chosen, that the LOS Model Correction and the LOS Projection and Gridding algorithms have been executed to create a TIRS LOS projection grid file. The L1T image may be in either the SCA-separated or SCA-combined format, as noted above, and would use either the SOM or UTM path-oriented projection. In any case, the TIRS spectral bands would be resampled to 30m pixel spacing.

7.3.6.3 Inputs

The BRAA and its component sub-algorithms use the inputs listed in the following table. Note that some of these "inputs" are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data).

Algorithm Inputs
ODL file (implementation)
Calibration Parameter File name
TIRS L1T image file name(see maturity note #3)
OLI L1T image file name (optional)
TIRS LOS projection grid (optional)
B2B characterization output file
Output residuals file name
Output statistics file name
TIRS bands to process
OLI bands to process (optional)
Processing Parameters
Outlier (t-distribution) threshold
Tie-point type (1 = regularly spaced, 2 = selected using TIRS grid)
Tie-point spacing in line direction
Tie-point spacing in sample direction
Fill range maximum
Fill range minimum
Fill threshold or percentage
Correlation window size lines
Correlation window size samples
Trending flag
Geometric Characterization ID (for trending)
Work Order ID (for trending)
WRS Path (for trending)
WRS Row (for trending)
Calibration Parameter File
Fill range maximum (default value)
Fill range minimum (default value)
Fill threshold or percentage (default value)
Correlation window size lines (default value)
Correlation window size samples (default value)
Maximum allowable offset
Strength of correlation peak
Correlation Fit method
Trending metrics – standard deviation thresholds per band (see note #3).

7.3.6.4 Outputs

Pan downsampled image (if OLI bands are included)
B2B residuals file (see Table 2 below for details)

B2B output data file (see Table 1 below for details)
B2B statistics file (see Table 3 below for details)
B2B characterization trending (if trend flag set to yes, see Table 3 for details)

7.3.6.5 Options

Grid-based tie-point generation
SCA-separated TIRS-only processing or SCA-combined OLI/TIRS processing

7.3.6.6 Prototype Code

Input to the executable is an ODL file; output is a set of ASCII files containing measured offsets between band locations with and without SCAs combined.

The prototype code was compiled with the following options when creating the test data files:
-g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The trending option was not included in the TIRS implementation since it does not affect the characterization algorithms and has been implemented in the OLI version of B2B. Since there had to be a change with how bands are differentiated between OLI and TIRS to keep track of the respective metadata, there have been minor changes to many of the TIRS executables that have been previously delivered (i.e., oli_grid, makegeomgrid, geomresample, oli_resamp, and oliresample) and to some library modules.

The enumerated band numbers are currently defined as:
{berror=-1, t1a, t1b, t2a, t2b, pan, b01, b02, b03, b04, b05, b06, b07, b09, b12, b13, b16, b17, b18, NBANDS, OPTICAL_AXIS}.
The TIRS modules, at this point in time, are only concerned with the bands through b09 in the above enumeration list. This equates to band numbers 1 to 9 for OLI and bands 10, 11, 14, and 15 for TIRS. The bands and their respective metadata and grids are now kept track of with the use of a band look up table.

The following text is a brief description of the main set of modules used within the prototype with each module listed along with a very short description. It should be noted that almost all library modules are not referenced in the explanations below. The modules within the main b2bchar directory of the prototype are discussed and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

b2b_char
Main driver for application. Calls routines to retrieve ODL input and CPF parameters, read and verify image metadata, reduce resolution of OLI PAN band, create a set of tiepoints, and calls module that will perform correlation on image tiepoint locations. Separate calls are made for creating tiepoints depending on whether points are to be evenly spaced or based upon a resampling grid.

get_parms
Reads input ODL parameters. Checks validity of input band combinations listed in ODL file. Reads CPF BRAA processing parameters.

verify_band_combos

Verifies search and reference band combinations given as input. Verification is done by matching reference and search band list. If bands given don't match, an error is returned.

create_tiepoints
Driver for creating evenly spaced tiepoints. Calls det_tiepoints for each band combination storing tiepoint locations in GCPLIB data structure.

det_tiepoints
Calculates a set of evenly spaced tiepoint locations based on image size. Tie points are based on number of points given as an input ODL parameter and the size of the image file.

create_tiepoints_grid
Driver for creating tie points based on the resampling grid.

downsample
Main driver for reducing the resolution of the OLI PAN band. Driver calls modules to initialize reduce image file (setup_reduce_img), calculates cubic convolution weights (cubic_convolution_weights), and applies cubic convolution weights to the PAN band (reduce).

setup_reduce_img
Initializes PAN reduced image file creation.

cubic_convolution_weight
Determines cubic convolution weights.

reduce
Applies cubic weights to PAN band. Output is written to file created/initialized in setup_reduce_img.

b2b_corr
Main driver for band correlation, or band mensuration, process. Driver opens image, calls module to perform correlation at tie-point locations (process_gcp), and writes out band registration residuals file (table 1). The majority of the OLI/TIRS band differentiation logic is contained here. process_gcp is called on for each SCA and band combination given in the input ODL file.

process_gcp
Process to perform correlation between two bands for one SCA. See Ground Control Point Correlation ADD for information on the LDCM correlation modules and process. Calls module xxx_check_fill to determine if a given window of imagery contains enough "non-fill" pixels so that mensuration can be performed.

ias_math_check_pixels_in_range
Checks to see if percent of pixels within a given buffer contains fill. Fill is passed in as a parameter. Module has been modified so that fill is a range rather than a single value.

math_fine_corr
Math library routine that implements the new (see below) least squares correlation algorithm developed for fine sub-pixel offset measurement. Takes same-size reference and search image windows as input and returns measured offsets.

7.3.6.7 Procedure

TIRS Band Registration Accuracy Assessment measures the misalignment between the TIRS spectral bands and, optionally, between the TIRS and OLI spectral bands, after all known geometric effects have been taken into account. In the case where an SCA-separated TIRS-only image is used as input, the results from the band registration assessment can be used by the TIRS band alignment calibration routine (See TIRS Band Alignment Calibration ADD) to estimate new Legendre LOSs (See TIRS Line-of-Sight Model Creation ADD) for both TIRS bands for each SCA. If an SCA-combined TIRS and OLI image is used as input, the results would be used solely for band registration accuracy characterization purposes. Due to the different viewing angles for each band of each SCA within TIRS, and between TIRS and OLI, geometric displacement due to relief must be removed from the imagery for band-to-band characterization of Earth acquisitions, i.e. input imagery for band registration assessment must be precision and terrain corrected (See TIRS Resampling ADD).

The steps involved in band registration assessment are depicted in Figure 1 and include creating data sets with common pixel resolutions (if the OLI panchromatic band is included); choosing locations (tie-point locations) for measurement; performing mensuration; removing outliers from calculated residuals; and calculating statistics from the remaining residuals. Residuals are measured on each SCA for each band combination requested through the input parameters.

7.3.6.7.1.1 Stage 1 - Data Input

The data input stage involves loading the information required to perform the band registration assessment. This includes reading the image file, retrieving the output B2B file names: output, residuals and statistic files; retrieving or initializing processing parameters: maximum displacement, fill range, fill threshold, minimum correlation peak, t-distribution threshold, bands to process, correlation window size, trending metrics, tie-point method; and if tie-point method is set to grid-based (for SCA-separated input images) the TIRS LOS grid file name will be read. Once the input file, and if need be the TIRS LOS grid name, has been retrieved the files and the information stored within them can be opened and read.

7.3.6.7.1.2 Optional Stage 2 - Creating a Reduced Resolution PAN band (if OLI and TIRS SCA-combined image is input)

Before displacement between the OLI PAN band and the other multispectral bands can be measured the PAN band must be reduced in resolution to match that of the multispectral bands. An oversampled cubic convolution function is used to reduce the resolution of the PAN band. Cubic convolution interpolation uses a set of piecewise cubic spline interpolating polynomials. The polynomials have the following form:

$$f(x) = \begin{cases} (\alpha+2)|x|^3 - (\alpha+3)|x|^2 + 1 & 0 \le |x| < 1 \\ \alpha|x|^3 - 5\alpha|x|^2 + 8\alpha|x| - 4\alpha & 1 \le |x| < 2 \\ 0 & |x| > 2 \end{cases}$$

Since the cubic convolution function is a separable function, a two dimensional representation of the function is given by multiplying two one-dimension cubic convolution functions, one function representing the x-direction the other function representing the y-direction. For an offset of zero, or x

= 0, and α = -1.0 the discrete cubic function has the following values; f(0) = 1 and f(n) = 0 elsewhere. Thus convolving the cubic convolution function of x = 0 with a data set leaves the data set unchanged.

$$y[n] = f[n] \otimes x[n]$$
$$\text{for } x = 0$$
$$\text{gives } y[n] = x[n]$$
$$\text{where } \otimes \text{ is the convolution operator}$$

Figure 2 shows what the cubic function f(t) (dashed line) and the corresponding discrete weights for an offset, or phase, of zero (crossed-dots).

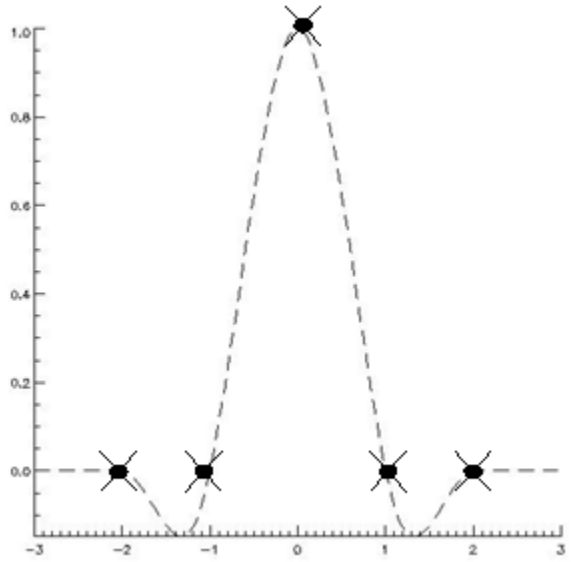


Figure 2. Cubic Convolution Function and Weights for phase of zero.

To spatially scale an input data stream an oversampled cubic convolution function with a offset of x =0 can be used. This can best be understood by looking at the Fourier Transform scaling property of a function that is convolved with a given input data set:

$$f(t) \otimes x(t) \Leftrightarrow F(\omega) \bullet X(\omega)$$

$$x(at) = \frac{1}{|a|} X\left(\frac{\omega}{a}\right)$$

Where:

$\otimes$ is convolution
● is multiplication
F is the Fourier transform of *f*
X is the Fourier transform of *x*
t is time
ω is frequency

Applying the cubic function and scaling properties to an image data file shows that densifying the points applied with the cubic convolution function will in turn inversely scale the function in the frequency domain, thus reducing the resolution of the imagery. By setting the cubic convolution offset to zero, densifying the number of weights of the cubic function, and convolving these weights to an image file a reduction in resolution will be the resultant output image file. Figure 3 shows the cubic function with corresponding weights densified by a factor of two and a phase shift of zero. To ensure that the cubic weights do not scale the DNs of the output imagery during convolution the cubic weights are divided by the scale factor.

$$f_s[n] = \frac{1}{2}\sum_{n=-4}^{4} f\left(\frac{n}{2}\right)$$

Where:

f_S[n] = scaled cubic convolution weights
f(n) = cubic convolution function

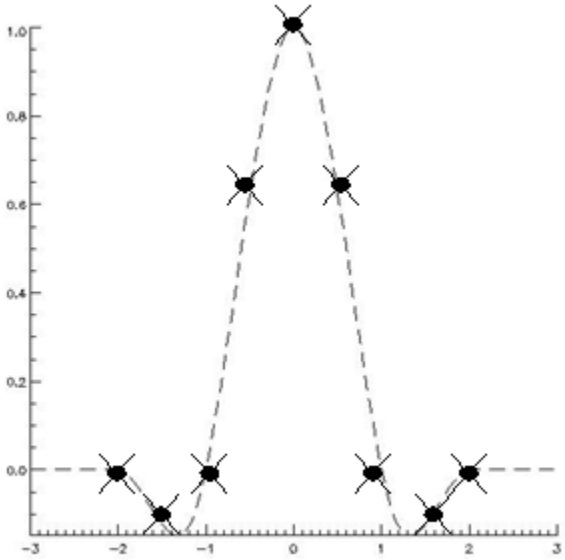


Figure 3. Cubic Convolution Densifyied by a factor of 2

Scaling the cubic convolution function by a factor of 2 gives the following 1-dimensional set of weights:

$$ccw[n] = \begin{bmatrix} 0.0 & -0.0625 & 0.0 & 0.3125 & 0.5 & 0.3125 & 0.0 & -0.0625 & 0.0 \end{bmatrix}$$

To determine the 2-dimensional cubic convolution weights two 1-dimensional sets of cubic weights are multiplied together (note only 7 values are needed for ccw, outside of this extent the weights are zero):

$$ccw[n] \times ccw[m] = ccw[n,m] = \begin{bmatrix} 0.0039 & 0.0 & -0.0195 & -0.0313 & -0.0195 & 0.0 & 0.0039 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.0195 & 0.0 & 0.0977 & 0.1563 & 0.0977 & 0.0 & -0.0195 \\ -0.0313 & 0.0 & 0.1563 & 0.25 & 0.1563 & 0.0 & -0.0313 \\ -0.0195 & 0.0 & 0.0977 & 0.1563 & 0.0977 & 0.0 & -0.0195 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0039 & 0.0 & -0.0195 & -0.0313 & -0.0195 & 0.0 & 0.0039 \end{bmatrix}$$

Where:

ccw[n] is a 8x1 1-dimensional set of cubic weights
ccw[m] is a 1x8 1-dimensional set of cubic weights

7.3.6.7.1.2.1 Procedure for Reducing PAN band

To reduce the resolution of the OLI PAN band apply the ccw[n,m] weights to the PAN image data:

$$\text{reduced pan} = ccw[n,m] * \text{pan band}$$

Note: number of lines and number of samples listed below pertain to the size of the PAN band imagery.

Reduce PAN Band Resolution Processing Steps

2. Set line =0 then for every other PAN line

   2.1. Set sample = 0 then for every other PAN sample

   2.2. initialize summing variable sum = 0.0

   2.3. For m = -4 to 4

      2.3.1. For n = -4 to 4

      2.3.2. Check to see if current image index is within valid imagery

      2.3.3. if m + line < 0 then line index = -m - line
      else if m + line >= number of lines then line index =
      2 * number of lines - m - line - 1
      else line index = m + line

      2.3.4. if n + sample < 0 then sample index = -n - sample
      else if n + sample >= number of sample then sample index =
2 * number of samples - n - sample - 1
      else sample index = n + sample

      2.3.5. sum = sum + ccw[n+4,m+4] • pan[line index, sample index]

2.4. Store output DN for reduced PAN
output line = line / 2
output sample = sample / 2
reduce pan[output line,output sample] = sum

7.3.6.7.1.3 Stage 3 - Create Tie-point Locations

Tie point locations may be determined in an evenly spaced pattern in output space or they may be established in an evenly spaced pattern in input space, using the TIRS LOS projection grid. The first method is used when SCA-combined OLI/TIRS images are input, and the second is used when SCA-separated TIRS-only images are input.

7.3.6.7.1.3.1 Determine Evenly Spaced Tie-points (See notes #6 and #7)

To determine evenly spaced tie-point locations a tie-point location is defined by stepping through the output space of the imagery by the user defined steps N,M.

Create Evenly Spaced Tie-Points Processing Steps

3. Determine number of tie-points in sample and line direction:

$$\text{tie - point spacing x} = \frac{\text{ONS} - \text{correlation window samples}}{\text{M - 1}}$$

$$\text{tie - point spacing y} = \frac{\text{ONL} - \text{correlation window lines}}{\text{N - 1}}$$

Where:
M = user entered number of tie-points in sample direction
N = user entered number of tie-points in line direction
ONS = number of samples in output space of multispectral band
ONL = number of lines in output space of multispectral band
Correlation window samples = user entered correlation window size in samples
Correlation window lines = user entered correlation window size in lines

4. Set evenly spaced tie-point locations
4.1. For j = 0 to N-2

$$\text{tie - point location y}[\text{j}] = \frac{\text{correlation window lines}}{2} + \text{j} * \text{tie - point spacing y}$$

4.2. $$\text{tie - point location y}[\text{N} - 1] = ONL - \frac{\text{correlation window lines}}{2}$$

4.3. For i = 0 to M-2

$$\text{tie - point location x}[\text{i}] = \frac{\text{correlation window samples}}{2} + \text{i} * \text{tie - point spacing x}$$

4.4. $$\text{tie - point location x}[\text{M - 1}] = ONS - \frac{\text{correlation window samples}}{2}$$

7.3.6.7.1.3.2 Determine TIRS Grid Spaced Tie-points (See notes #6 and #7)

For descriptions of the format and data stored within the TIRS LOS grid see the TIRS Line of Sight Projection to Ellipsoid and Terrain ADD.

Input Space Tie-points Processing steps.

1. Read image extent parameters from TIRS LOS grid
   INS = input (raw) space number of samples
   INL = input (raw) space number of lines

2. Determine number of tie-points in sample and line direction:

$$\text{spacing x} = \frac{\text{INS} - \text{correlation window samples}}{\text{M - 1}}$$

$$\text{spacing y} = \frac{\text{INL} - \text{correlation window lines}}{\text{N - 1}}$$

3. Establish input (raw) space tie-point locations

   3.1 For j = 0 to N-2

$$\text{y}[\text{j}] = \frac{\text{correlation window lines}}{2} + \text{j} * \text{spacing y}$$

   3.2 $\text{y}[\text{N} - 1] = INL - \frac{\text{correlation window lines}}{2}$

   3.3 For i = 0 to M-2

$$\text{x}[\text{i}] = \frac{\text{correlation window samples}}{2} + \text{i} * \text{spacing x}$$

   3.4 $\text{x}[\text{M - 1}] = INS - \frac{\text{correlation window samples}}{2}$

4. Project inputs space tie-points locations to output space

   4.1 For j=N-1

      4.1.1 For i=M-1
      Map input space tie-point location to output space using grid mapping coefficients.
      tie-point location y = $b_0 + b_1 * x[i] + b_2 * y[j] + b_3 * x[i] * y[j]$
      tie-point location x = $a_0 + a_1 * x[i] + a_2 * y[j] + a_3 * x[i] * y[j]$

Where (See note #7):
a_n = forward sample mapping coefficients for zero elevation plane retrieved from the TIRS LOS projection grid
b_n = forward line mapping coefficients for zero elevation plane retrieved from the TIRS LOS projection grid

7.3.6.7.1.4 Stage 4. Calculate Individual Point-by-Point Band Displacements

Normalized cross correlation is used to measure spatial differences between the reference and search windows extracted from the two bands being compared. The normalized cross correlation process helps to reduce any correlation artifacts that may arise from radiometric differences between the two image sources. The correlation process will only measure linear distortions over the windowed areas. By choosing appropriate correlation windows that are well distributed throughout the imagery, nonlinear differences between the image sources can be found. Normalized grey scale correlation has the following formula:

$$R(x,y) = \frac{\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2}\left[\left(f(j,i)-\bar{f}\right)\left(g(x+j,y+i)-\bar{g}\right)\right]}{\left[\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2}\left(f(j,i)-\bar{f}\right)^2\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2}\left(g(x+j,y+i)-\bar{g}\right)^2\right]^{1/2}}$$

Where:

N = M = Correlation window size in lines and samples
R = correlation surface (N,M) (See note# 10)
f = reference window (N,M)
g = search window (N,M)

$$\bar{f} = \frac{1}{(M+1)(N+1)}\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2} f(j,i)$$

$$\bar{g} = \frac{1}{(M+1)(N+1)}\sum_{j=-N/2}^{N/2}\sum_{i=-M/2}^{M/2} g(x+j,y+i)$$

Normalized cross correlation will produce a discrete correlation surface (i.e., correlation values at integer x,y locations). A sub pixel location associated with the displacement is found by fitting a polynomial around a 3x3 area centered on the correlation peak. The polynomial coefficients can be used to solve for the peak or sub pixel location. Once the discrete correlation has been calculated and the peak value within these discrete values has been found the sub-pixel location can be calculated:

$$P(y,x) = a_0 + a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2$$

Where
P(x,y) is polynomial peak fit
x = sample direction
y = line direction

Set up matrices for least squares fit of discrete R(x,y) to x/y locations.

$$\underset{9x1}{\begin{bmatrix} R(-1,-1) \\ R(-1,0) \\ \vdots \\ R(1,0) \\ R(1,1) \end{bmatrix}} = \underset{9x6}{\begin{bmatrix} 1 & -1 & -1 & (-1)*(-1) & (-1)^2 & (-1)^2 \\ 1 & 0 & -1 & (-1)*(0) & (0)^2 & (-1)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 1 & (1)*(0) & (0)^2 & (0)^2 \\ 1 & 1 & 1 & (1)*(1) & (1)^2 & (1)^2 \end{bmatrix}} \underset{6x1}{[a]}$$

or: [Y] = [X] [a]

Note that R(x,y) is relative to the peak, the total offset will need to have the integer line offset and sample offset added to the sub-pixel location to have the total measured offset. Solving for the peak polynomial using least squares:

$$[a] = \left([X]^T [X]\right)^{-1} [X]^T [Y]$$

Calculating the partial derivative of P(x,y) in both the x and y directions, setting the partial equations to zero, and solving the partials for x and y, gives the sub-pixel location within the sub-pixel 3x3 window.

$$\frac{\partial}{\partial x} P(x, y) = a_1 + a_3 y + 2a_4 x = 0$$
$$\frac{\partial}{\partial y} P(x, y) = a_2 + a_3 x + 2a_5 y = 0$$

Set partial equations equal to zero and solve for x and y:

$$\text{Sub-pixel x offset} = \frac{2a_1 a_5 - a_2 a_3}{a_3^2 - 4a_4 a_5}$$

$$\text{Sub-pixel y offset} = \frac{2a_2 a_4 - a_1 a_3}{a_3^2 - 4a_4 a_5}$$

The steps for mensuration, calculating the total offset measured, and how they fit in the overall procedure is given in the processing steps section.

See Ground Control Correlation ADD for prototype specifications of correlation processes.

7.3.6.7.1.5 Stage 5. Removing Outliers Using the t-distribution

Once all the line and sample offsets have been measured and the first level of outlier rejection has been performed, a check against the maximum allowable offset and the minimum allowable correlation peak, the measurements are further reduced of outliers using a Student-t outlier rejection.

Given a t-distribution tolerance value, outliers are removed within the data set until all values deemed as “non-outliers” or “valid” fall inside the confidence interval of a t-distribution. The tolerance, or associated confidence interval, is specified per run (or processing flow) and usually lies between 0.9-0.99. The default value is 0.95. The number of degrees of freedom of the data set is equal to the number of valid data points minus one. The steps involved in this outlier procedure are given below. The process listed works on lines and samples simultaneously, calculating statistics independently for each.

Student-t Outlier Rejection Processing steps.
If an SCA-separated image is being analyzed, the outlier rejection sequence is executed independently on each SCA. For SCA-combined images, the outlier rejection logic is performed for all points at once.

For each SCA:

1. Calculate mean and standard deviation of data for lines and samples (see stage #6).

$$\text{mean offset} = \frac{1}{N}\sum_{i=0}^{N}\text{offset}_i$$

$$\text{standard deviation} = \frac{1}{N-1}\sum_{i=0}^{N}\left(\text{offset}_i - \text{mean offset}\right)^2$$

Where:
N = number of valid offsets measured (above peak threshold and below maximum offset)

Two means and standard deviations are calculated, one for the line direction and one for the sample direction.

2. Find largest offset and compare it to outlier threshold.

   2.1. Calculate two tailed t-distribution (T) value for current degree of freedom (N-1) and confidence level α.

   2.2. Calculate largest deviation from the mean allowable for the specified degree of freedom and α:
   $\Delta line = \sigma_{line} * T$
   $\Delta sample = \sigma_{sample} * T$
   Where:
   σ_{line} = standard deviation of valid line offsets
   σ_{sample} = standard deviation of valid sample offsets

   2.3. Find valid data point that is farthest from the mean.
   $\max line_i = MAX\{$ line offset - mean line offset$\}$
   $\max sample_j = MAX\{$ sample offset - mean sample offset$\}$
   Where:
   The maximum is found from all valid offsets
   i is the tie-point number of max line
   j is the tie-point number of max sample

   2.4. If valid data point that is farthest from the mean is greater than the allowable Δ then the valid point is flagged as outlier.
   if $\max line_i > \Delta line$ or $\max sample_j > \Delta sample$ then
   if($\max sample_j / \sigma_{sample} > \max line_i / \sigma_{line}$)
   tie-point j is marked as an outlier
   else
   tie-point i is marked as an outlier
   else no outliers found

3. Repeat 1 and 2 above until no outliers are found.

The t-distribution outlier detection prototype logic runs as a separate process. The key components of this process include:

tdist
Main driver for t-distribution outlier rejection and for band residuals statistic calculations. Logic to report the SCA to which each set of statistics pertains is included. SCA 0 will be used to indicate measurements from SCA-combined images. Calls modules to read input parameters (getpar), read tie-point information (get_gcpdata), perform t-distribution outlier rejection (filter_outliers), calculate final band registration statistics (put_gcpdata), and to write final band registration results (put_gcpdata). The t-distribution outlier rejection is a separate step in the current prototype code, whether the process is separated from the BRAA process or is combined within as a function call does not affect the algorithm.

getpar
Reads ODL input parameters file.

get_gcpdata
Reads output file created from B2B characterization (See Table 1). Tie-point and mensuration information is stored in data structure and passed into filter_outliers module.

filter_outliers
Performs t-distribution outlier rejection. Input is the tie-point data structure populated from get_gcpdata. Outliers are flagged and reported in the band registration residuals file.

put_gcpdata
Calculates statistics of final residuals and prints results. Printed results are shown in Tables 2 and 3. Calls modules to perform heap sort on statistics and to compute median of residuals calculated.

7.3.6.7.1.6 Stage 6. Calculating Measured Statistics

The mean, standard deviation, minimum, maximum, median, and root-mean squared offset (RMS) are calculated from the tie-points that pass all outlier criteria; below maximum offset, above peak threshold, and student t-distribution test. The calculation for mean, standard deviation, and RMS are shown below where x_i is the measured offset.

mean: $m_x = \frac{\sum_{i=0}^{N-1} x_i}{N}$

standard deviation: $\sigma_x = \sqrt{\frac{\sum_{i=0}^{N-1} (x_i - m_x)^2}{N-1}}$

RMS: $RMS_x = \sqrt{\frac{\sum_{i=0}^{N-1} (x_i)^2}{N}}$

7.3.6.7.1.7 Band Accuracy Assessment Processing steps

Windows extracted from imagery have the user entered dimensions; correlation window lines and correlation window samples. Correlation parameters have been read or set as default values; maximum offset, fit method, correlation peak, fill data range, fill threshold. The bands should be indexed so that the TIRS 10.8 micrometer band is used as a reference to all other bands.

1. For SCA = Number of SCAs to process (1 if SCA-combined, 3 if SCA-separated)

    1.1. For rband = Total number of TIRS and OLI bands to process

        if rband is equal to OLI PAN use reduced PAN image file

    1.2. For sband = rband + 1 to Total number of TIRS and OLI bands to process

    1.3. For index = Number of tie-points to process

        1.3.1. Read current tie-point chip and tie-point location x,y
        Set tie-point flag to unsuccessful

        1.3.2. Extract sband window (of imagery) at tie-point location x,y

        1.3.3. Extract rband window (of imagery) at tie-point location x,y

        1.3.4. Count number of pixels in rband window that is within fill range.
        count = 0
        For i=0 to number of pixels in correlation window
            If rband pixel is > fill min and rband pixel is < fill max
                count++

        1.3.5. Check number of rband pixels counted against fill threshold/percentage.
        $$\text{if } \frac{\text{count}}{\text{correlation window size}} > \text{fill threshold}$$
                increment index to next tie-point location
            else
                continue

        1.3.6. Count number of pixels in sband window that is within fill range.
        count = 0
        For i=0 to number of pixels in correlation window
            If sband pixel is > fill min and sand pixel is < fill max
                count++

        1.3.7. Check number of sbands pixels counted against fill threshold/precentage.
        $$\text{if } \frac{\text{count}}{\text{correlation window size}} > \text{fill threshold}$$
                increment index to next tie-point location
            else
                continue

1.3.8. Perform normalized grey scaled correlation between rband and sband windowed images, calculating correlation surface *R* (See Stage 4 and notes #9 and #10).

1.3.9. Find peak within correlation surface

```
Max = R(0,0)
For i=0 to correlation window number of lines -1
        For j=0 to correlation window number of samples -1
                If R(i,j) > max then
                        Max = R(i,j)
                        line offset = i
                        sample offset = j
```

1.3.10. Check correlation peak against threshold

```
if max > peak threshold
        continue
else
        set tie-point flag to outlier and choose next tie-point
```

1.3.11. Measure sub-pixel peak location (see stage #4)

Δsub-line
Δsub-sample

1.3.12. Calculate total pixel offset

total line offset = line offset + Δsub-line
total sample offset = sample offset + Δsub-sample

1.3.13. Check offset against maximum displacement offset

$$\text{total displacement} = \sqrt{(\text{total line offset})^2 + (\text{total sample offset})^2}$$

if (total displacement > maximum displacement)
        Set tie-point flag to outlier and choose next tie-point
Else
        Set tie-point flag to valid

1.4. Store SCA and band combination (rband-to-sband) tie-point mensuration information, correlation success, and offsets measured. See table #1.

2. For SCA = 1 to Number of SCAs to process

2.1. For band combination = 1 to Number of band combinations

2.1.1. Perform t-distribution outlier rejection (See stage #5).

2.2. Store SCA and band combination final individual tie-point information and outlier flag. See table #2.

3. For SCA = 1 to Number of SCAs to process

4. For band combination = 1 to Number of band combinations

   4.1. Calculate mean, minimum, maximum, median, standard deviation, and root mean squared offset.

   4.2. Store SCA and band combination statistics. See table #3.

5. Perform trending if trending flag is set to yes

   5.1. Check results against trending metrics
   For each band of each SCA
   if measured Standard Deviation > trending metric
   exit trending
   If there are no Standard Deviation > trending metric perform trending

7.3.6.8 Output files

The output files listed below for the TIRS BRAA follow the philosophy of the Advanced Land Imagery Image Assessment System (ALIAS) Band-to-Band (B2B) Characterization output files in that they are made generic so that the same format can be used elsewhere. Therefore some fields like latitude, longitude, and elevation may not apply to the application and would be filled with zeros or nominal values.

All output files contain a standard header. This standard header is at the beginning of the file and contains the following:

1) Date and time file was created.
2) Spacecraft and instrument(s) pertaining to measurements.
3) Off-nadir (roll) angle of spacecraft/instrument.
4) Acquisition type
5) Report type (band-to-band)
6) Work order ID of process (left blank if not applicable)
7) WRS path/row
8) Software version that produced report.
9) L0R image file name

The data shown within Table 3 listed below is stored in the database. Database output will only be performed if the standard deviation statistics for every band and every SCA are below the thresholds (trending metrics) stored in the CPF (see note #3). The statistics stored per band per SCA will be used for trending analysis of the band registration accuracy of the TIRS instrument. Results produced through a time-series analysis of this data stored, over a set time interval or multiple image files, will determine if new Line-of-Sight (LOS) Legendre coefficients will need to be generated from the TIRS Band-to-Band Calibration Algorithm (See TIRS Band-to-Band Calibration ADD for details). These statistics may also be needed for providing feedback to the LDCM user community about the band registration of LDCM products generated.

Field	Description
Date and time	Date (day of week, month, day of month, year) and time of file creation.
Spacecraft and instrument source	LDCM and TIRS (and OLI if applicable)
Processing Center	EROS Data Center SVT
Work order ID	Work order ID associated with processing (blank if not applicable)
WRS path/row	WRS path and row (See note #11)
Software version	Software version used to create report
Off-nadir angle	Off-nadir roll angle of processed image file (See note #12)
Acquisition Type	Earth viewing (for TIRS)
L0R image file	L0R image file name used to create L1T
Processed image file name	Name of L1T used to create report
Reference bands	Reference bands used in band assessment
Search bands	Search bands used in band assessment
Heading for individual tie-points	One line of ASCII text defining individual tie-point fields.
For each tie-point:	
Tie point number	Tie-point index/number in total tie-point list
Reference line	Tie-point line location in reference image (band)
Reference sample	Tie-point sample location in reference image (band)
Reference latitude	Tie-point latitude location
Reference longitude	Tie-point longitude location
Reference elevation	Elevation of tie-point location (see note #13)
Search line	Tie-point line location in search image
Search sample	Tie-point sample location in search image
Delta line	Measured offset in line direction
Delta sample	Measured offset in sample direction
Outlier flag	1=Valid, 0=Outlier
Reference band	Reference band number
Search band	Search band number
Reference SCA	SCA number that reference window was extracted or 0 if an SCA-combined L1T image was used
Search SCA	SCA number that search window was extracted or 0 if an SCA-combined L1T image was used
Search image	Name of search image
Reference image	Name of reference image

Table 9. Band Registration Accuracy Assessment Data File

Field	Description
Date and time	Date (day of week, month, day of month, year) and time of file creation.
Spacecraft and instrument source	LDCM and TIRS (and OLI if applicable)
Processing Center	EROS Data Center SVT
Work order ID	Work order ID associated with processing (blank if not applicable)
WRS path/row	WRS path and row (See note #11)
Software version	Software version used to create report
Off-nadir angle	Off-nadir pointing angle of processed image file (See note #12)
Acquisition Type	Earth viewing for TIRS
L0R image file	L0R image file name used to create L1T
Processed image file name	Name of L1T used to create report
Number of records	Total number of tie-points stored in file
Heading for individual tie-points	One line of ASCII text defining individual tie-point fields.
For each band combination	
Combination header	Number of points in combination, reference band number, search band number.
For each tie-point:	
Tie point number	Tie-point index/number in total tie-point list
Reference line	Tie-point line location in reference image (band)
Reference sample	Tie-point sample location in reference image (band)
Reference latitude	Tie-point latitude location
Reference longitude	Tie-point longitude location
Reference elevation	Elevation of tie-point location
Search line	Tie-point line location in search image
Search sample	Tie-point sample location in search image
Delta line	Measured offset in line direction
Delta sample	Measured offset in sample direction
Outlier flag	1=Valid, 0=Outlier
Reference band	Reference band number
Search band	Search band number
Reference SCA	SCA number that reference window was extracted from or 0 if an SCA-combined L1T input image was used
Search SCA	SCA number that search window was extracted from or 0 if an SCA-combined L1T input image was used
Search image	Name of search image
Reference image	Name of reference image

Table 10. Band Registration Accuracy Assessment Residuals File

Field	Description	Database Output
Date and time	Date (day of week, month, day of month, year) and time of file creation.	Yes
Spacecraft and instrument source	LDCM and TIRS (and OLI if applicable)	Yes
Processing Center	EROS Data Center SVT	Yes
Work order ID	Work order ID associated with processing (blank if not applicable)	Yes
WRS path/row	WRS path and row (See note #12)	Yes
Software version	Software version used to create report	Yes
Off-nadir angle	Off-nadir pointing angle of processed image file (See note #13)	Yes
Acquisition Type	Earth viewing for TIRS	Yes
L0R image file	L0R image file name used to create L1T	Yes
Processed image file name	Name of L1T used to create report	No
t-distribution threshold	Threshold used in t-distribution outlier rejection	Yes
For each band combination of each SCA processed		
Reference band	Reference band of statistics	Yes
Search band	Search band of statistics	Yes
SCA	SCA number of statistics or 0 if SCA-combined L1T input image was used	Yes
Total tie-points	Total number of tie-points for band combination of SCA	Yes
Correlated tie-points	Number of tie-points that successfully correlated for band combination of SCA	Yes
Valid tie-points	Total number of valid tie-points for band combination of SCA after all outlier rejection has been performed	Yes
For both line and sample direction:	All statistics are given in terms of pixels	
Minimum offset	Minimum offset within all valid offsets	Yes
Mean offset	Mean offset of all valid offsets	Yes
Maximum offset	Maximum offset within all valid offsets	Yes
Median offset	Median offset within all valid offsets	Yes
Standard deviation	Standard deviation of all valid offsets	Yes
Root-mean-squared	Root mean squared offset of all valid offsets	Yes

Table 11. Band Registration Accuracy Assessment Statistics Output File

7.3.6.8.1 Assessing Band Registration (Accessing Statistics Stored in Database)

The Band Accuracy Assessment statistics stored in the database will need to be accessed by the geometric CalVal team. Delineation, or essentially data base querying, will be done by the following or a combination of the following:

1) Date range of image acquisition or processing date
2) By SCA number (including 0 for SCA-combined results)
3) By band number
4) By acquisition type (Nadir, off-nadir, Lunar)
5) By geographic location of image extent.

At a minimum access to the Band Accuracy Assessment statistics is needed. Simple tools, such as an SQL queries, would be beneficial to the geometric CalVal team but are not absolutely necessary as they could be developed later through other means.

7.3.6.9 Maturity

1. The prototype implementation operates on separate TIRS and, if present, OLI L1T input image files. The logic would have to be adapted to an OLI-TIRS combined L1T image organization/structure to be used with an integrated product generation system.

7.3.6.10 Notes

Some additional background assumptions and notes include:

1. Correlation parameters, minimum correlation peak and maximum offset, are stored and retrieved from the CPF.
2. Tools that analyze the query results will be needed to generate summary statistics: scene statistics, individual bands per SCA, SCA summary, band summary. These statistics would ultimately be provided to the user as summary statistics in an image quality assessment for the user community.
3. As shown in the input table, there will be metrics, based on calculated statistics, as to whether trending should be performed or not. This metric would be provided to avoid having garbage stored in the database. The metric values would be stored and retrieved from the CPF. There would be one metric per band per SCA which would be compared to the standard deviation statistics for each SCA. The criteria to check for trending are shown in section 5.1 of the Band Accuracy Assessment Processing steps section.
4. Band Accuracy statistics stored within the database will be accessed for analysis.
    a. *Accessed according to a specific date range.*
    b. *Accessed according to a specific band or SCA.*
    c. *Accessed according to a specific geographic location.*
    d. *Accessed according to acquisition type (nadir, off-nadir, lunar).*

    This data accessed will be retrieved and stored within a comma delimited file. The methodology used to access the database could be an SQL script. This SQL query code could be developed either by the IPE or outside of the IPE.
5. Data stored within the database will be accessed for time series analysis.
    a. Data would be pulled out by scene/SCA band pairs for a user-specified time period.
    b. Statistics over multiple scenes would be calculated per SCA and/or per band. Then combined them into the SCA and/or band average statistics.
    c. Results could be compared to the band registration spec. These results could serve as triggers to other events, i.e. new CPF generation and testing.
    d. Results could be used to verify conformance with product specifications.

    These calculations could be performed within the methodology used to access the data from the database (SQL script).
6. Tie-point locations could also be stored and used as projection Y and X coordinates. The appropriate conversions must be done when converting between projection coordinates and

line and sample locations when extracting image windows between bands. This transformation should also include any rotation due to path orientated projections.

7. The prototype code uses a library call that maps any input point with a given elevation to output space. For BRAA, the elevation for the mapping point is set to zero. Since for BRAA the reference and search output space are the same, output line/sample in output reference space should be line/sample in output search space.
8. The c and d parallax coefficients are needed for each band or each SCA for every grid cell point. Therefore if the coefficients were stored as arrays stacked by grid column and then grid row for a particular input pixel that fell within grid cell column N and grid cell row M the c and d coefficients needed for that pixel would be indexed by: index = (M * number of grid columns + N) * 2. The factor of 2 is due to the fact the parallax odd/even effects are mapped as linear therefore 2 coefficients are stored for each the odd and even pixels of a grid cell.
9. The grey scale correlation process, or surface, can be implemented using a Fast Fourier Transform (FFT).
10. The correlation surface could be smaller than the search window depending on the search area or maximum offset.
11. Any kind of "non-WRS" collect; like lunar, should have 000/000 listed as the path/row. This is not applicable for TIRS.
12. Pointing angle for lunar acquisitions would be 0.0. This is not applicable for TIRS.
13. This tie point residual file structure is also used for the image registration accuracy characterization algorithm so it includes fields that are not required for both algorithms. An example is the elevation field which is set to 0 for this algorithm.
14. The correlation result fit method defines the algorithm used to estimate the correlation peak location to sub-pixel accuracy. Only the quadratic surface fitting method described in this ADD is supported in the baseline algorithm. The least squares correlation technique is not used for TIRS band registration assessment. Since the source images are oversampled to 30 meter resolution, the robustness of the normalized gray scale correlation is more important than the sub-pixel precision of the least squares method.

7.4 Common Radiometry Algorithms

7.4.1 Dropped Frame Characterization

7.4.1.1 Background

All dropped “frames” (aka dropped “lines” for this ADD), need to be accounted for during Ingest System (IS) processing and Product Generation System (PGS) processing.

Dropped frames are typically expected to occur onboard the spacecraft. The instrument will create a CRC value at the end of each video line. Once the data is on the ground, ingest will perform a CRC check on these values. Any check failure will result in the dropped frame being filled with zeros, and the setting of the CRC flag to fail in the OLI line header dataset.
Dropped frame conditions may also occur if a file within a given interval is missing. Such intervals will be archived for later processing. If or when selected, the interval may be broken up into two “sub-intervals” or the data may be filled to complete the interval. In the latter case, ingest will flag these all these frames as zero filled in the OLI line header dataset.

This algorithm will check the Line Header data in the ancillary file for all intervals and output any dropped frames to a Labeled Mask (LM) aka Artifact Mask (AM) and database.

Note, while this algorithm describes all necessary parameters and functionality required for processing, no prototype code will be developed or delivered by the algorithm developer. Instead, coding will be performed by software developers based upon finalized detailed knowledge of the ancillary data file content and structure.

7.4.1.2 **Dependencies**

None

7.4.1.3 **Input**

Descriptions	**Level**	**Source**	**Type**
Line Header data	interval	Ancillary Data File	

7.4.1.4 **Output**

Descriptions	**Level**	**Target**	**Type**
Dropped Frames	N_{band} x N_{frame}	AM	Int
# of Dropped Frames	N_{band} x N_{Frame}	Db	Int

7.4.1.5 **Options**

None.

7.4.1.6 **Prototype Code**

None.

7.4.1.7 **Procedure**

1. Read the interval ancillary data file and extract the LineHeader Dataset
2. Verify CRC check and line fill information from the Line Status field.

```
{
For (band=0,band=9,band++)

Num_filllines = 0          ; /*line counter initialization*/
Int Linearray[9,Num_fillines] ;  /* array containing CRC  check*/
Int Filltotal[9];  /*array containing # of dropped frames per band*/
{
  For  (line = 0, line=intervalsize, line++)
          {
        If (bit8 == 1)              ;  /*CRC check successful*/
```

```
            {
            If  (bit2 ==  1)          ; /*line has fill*/
                  Linearray[band,line] = 1
                  num_filllines[band] = num_fillines+1
            }
         Else
            Printf("CRC check unsuccessful')
   }
   Filltotal[band] = num_fillines;
   }
```

3. Output linearray and flag each corresponding AM value
4. Output Fill total to DB for trending

Reference

1. "L0r DFCB Landsat 8 Operational Land Imager (OLI) Swath Data Format Control Book (DFCB) -Version 1.0m", Jan 2009

7.4.2 Impulse Noise Characterization

7.4.2.1 Background

Impulse noise (IN) is a randomly occurring aperiodic noise source that appears as a sample with signal value notably different (far in excess of that expected due to Total Noise) from values exhibited by its nearest neighbors. Sources of IN can include the following:

- 'bit flips,' i.e., an 'on' bit that is recorded as 'off' or vice versa, due to transmission/recording errors affecting the data stream
- 'single-event upsets' (SEUs), i.e., significant increases in the analog output signal produced by a detector due to the effects of charged particles striking the detector array. This effect tends to appear as a saturated pixel with the value (2^N-*1*) DN (where *N* is the radiometric resolution of the sensor data in bits), followed by a saturated pixel with the minimum possible DN value (0). Subsequent pixels tend to manifest the 'expected' DN levels.

To detect IN artifacts in Level 0 image and calibration data, this algorithm calculates the absolute difference between a pixel value and a median filter value computed using that pixel's nearest neighborhood. This difference is then compared to a threshold value determined from the surrounding pixel values, detector total noise level (Detector Noise) and minimum detectable IN (IN Limit). Though a linear filter is faster, a median filter is more robust at predicting the true signal value. The median filter is one-dimensional, applied in the image pixel column direction. The affected pixel locations and their values are recorded in the LMASK to allow exclusion from further radiometric processing.

This algorithm is run only on homogenous data sources, i.e. dark shutter bias or lunar data. In addition, the algorithm is run on night image, lamp and solar OLI data and blackbody TIRS data. The data are processed serially, detector-by-detector, band after band.

7.4.2.2 Inputs

Descriptions	Symbol	Units	Level	Source	Type
Level 0 data: - Dark Earth (OLI only) - Space			N_{bands} x N_{SCAs} x $N_{detectors}$		Int

- Shutter - Lamp (OLI only) - Solar (OLI only) - Lunar - Blackbody (TIRS only)	Q	DN	x N_{frames}		
Median Filter Width	MFW	Pixels	N_{bands} x N_{SCAs}	CPF	Int
Inoperable detectors			N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Int
Detector Noise		DN	N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Float
IN Limit		DN	N_{bands} x N_{SCAs}	CPF	Int
Dropped Frames			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int

7.4.2.3 Outputs

Descriptions	Symbol	Units	Level	Target	Type
Number of pixels affected by IN			N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Int
IN Pixel Locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Db/LM	Int
IN Pixel value		DN	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Report	Int
IN Neighbor-Pixel values	DNb, DNa	DN	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Report	Int

7.4.2.4 Options

A parameter (switch) in work order should provide a possibility to generate a report file, if desired.

7.4.2.5 Procedure

For each band, each detector (d):

1. Ignore pixels corresponding to inoperable detectors, as well as those flagged in LM as dropped frames.

2. Append (MFW/2 – 1) samples to the beginning and the end of the image, to enable characterization of the pixels at, or close to the beginning and end of the image (pixels for which the filter would fall off the edge of the image).
    a. Copy (MFW/2 – 1) samples starting from sample #2 of original image to the added samples at the beginning of image.
    b. Copy (MFW/2 – 1) samples ending in the second last sample of original image to the added samples at the end of image.
3. Compute Median-Filter Value (*Q'*) within the MFW for each pixel (i). The median filter is one-dimensional, applied in the image pixel column direction. For example, for MFW=5,

$$Q'_{d,i} = Median(Q_{d,i-2}, Q_{d,i-1}, Q_{d,i}, Q_{d,i+1}, Q_{d,i+2})$$

    a. To characterize pixels surrounding dropped frames, ignore dropped frames. If, for example, sample $Q_{d,i+1}$ were from a dropped frame, compute the Median-Filter value as.

$$Q'_{d,i} = Median(Q_{d,i-2}, Q_{d,i-1}, Q_{d,i}, Q_{d,i+2}, Q_{d,i+3})$$

4. Calculate the absolute difference between the pixel value (*Q*) and its *Q'* for each pixel (i)

$$A(i) = | Q(i) - Q' |$$

5. Calculate the absolute difference between DN values of the neighboring pixels.

$$Diff(i) = | Q(i+1) - Q(i-1) |$$

6. If Diff(i) is:
    a. greater than twice the Detector Noise, then

$$Threshhold = Diff(i) * IN\_Limit(SCA) / (2*Detector\_Noise(d))$$

    b. less than, or equal to twice the Detector Noise, then

$$Threshold = Detector\_Noise(d) * IN\_Limit(SCA)$$

7. Identify an occurrence of IN for each pixel where A(i) > Threshold and store in the database the number of pixels affected by IN. If the report file generation is selected as an option in the work order, store into the report file the scene type, band number, SCA number, detector number, frame number (i), and actual output values of the corrupted pixels and their neighbors.
8. Remove from image data the samples added in step 2
9. Update the label mask to include the pixels identified as corrupted by IN.

7.4.3 Saturated Pixel Characterization

7.4.3.1 Background

This algorithm flags pixels in data that have saturated digital counts. Two types of saturation can be observed in the OLI data: digital and analog. In TIRS data, at least one type (digital) is expected, but there might also exist a similarly defined analog saturation level. This algorithm is applicable whether one or both types of saturation should exist.

Digital saturation occurs when the analog input signal represents a radiance level outside of the range that the detector's Analog/Digital (A/D) converter can properly process. The result is that the corresponding digital output is set to 0 DN at the lower end (Digital Low Saturation) and (2^N-1) DN at the upper end (Digital High Saturation) of dynamic range, where N is the signal quantization in bits.

Analog High Saturation occurs when the input signal exceeds the maximum input radiance to which the instrument responds (approximately) linearly or according to any other predefined function. A similar situation occurs at low saturation levels, defining the Analog Low Saturation. The analog saturation levels are determined either from prelaunch measurements or on-orbit characterization and saved in CPF.

The saturated output counts have unknown corresponding radiances and using these data can lead to erroneous research results. Therefore, it is essential to flag saturated pixels for appropriate handling in subsequent processing.

7.4.3.2 Inputs

Descriptions	Symbol	Units	Level	Source	Type
All Level 0 data	Q	DN	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Int
Impulse Noise Locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Dropped Frame Locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Significant Bit Flag (least, 'L', or most, 'M', significant 12 bits)			1	metadata	Char
Analog Low Saturation Level		DN	N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Int
Analog High Saturation Level		DN	N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Int
Digital Low Saturation Level		DN	N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Int
Digital High Saturation Level		DN	N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Int

7.4.3.3 Outputs

Descriptions	Symbol	Units	Level	Source	Type

Digital Low Saturation Pixel Total			N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Int
Digital High Saturation Pixel Total			N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Int
Digital Low Saturation Pixel Locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Digital High Saturation Pixel Locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Analog Low Saturation Pixel Total			N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Int
Analog High Saturation Pixel Total			N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Int
Analog Low Saturation Pixel Locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM & Report	Int
Analog High Saturation Pixel Locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM & Report	Int
Analog Low Saturation Pixel Values		DN	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Report	Int
Analog High Saturation Pixel Values		DN	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Report	Int

7.4.3.4 Options

A parameter (switch) in work order provides the option to generate a report file.

7.4.3.5 Procedure

For each detector, d, including inoperable detectors, for all bands and SCAs:

1. Ignore pixels flagged in LM as dropped frames or corrupted with impulse noise. Thus, if a pixel identified as impulse noise affected has a value equal to saturation value, it will not be identified as saturated.
2. Using analog saturation levels from CPF, find and flag each saturated pixel. Analog saturated pixels are those pixels that have DN values below Analog Low Saturation Level, but higher than the Digital Low Saturation Level, and pixels that have DN values above Analog High

Saturation Level, but lower than the Digital High Saturation Level. Thus, if analog and digital saturation levels are set to the same value, no analog saturation will be detected. Use different flags for low and high saturation.

3. Using digital saturation levels, 0 DN for low and (2^N-1) DN for high saturation, find and flag all digitally saturated pixels. Use different flags for low and high saturation and different from flags used to identify analog saturation levels.
4. Output all flags to corresponding pixel locations in LM file
5. Record the number of low and high saturated pixels per detector to the database for both digital and analog types of saturation. Record pixel values from analog saturated pixels to the report file, if the report file is generated.

7.4.4 Histogram Statistics Characterization

7.4.4.1 Background

This algorithm serves as a general purpose algorithm occurring at different locations in the L1R processing flow. It supports characterization of all active detectors, including the inoperable ones, by computing statistics from single images or collects up to ~4.5 minutes long: minimum, maximum, mean, standard deviation, skewness, and kurtosis. In addition, the algorithm calculates for each detector the mean of squared pixel values and the adjacent detector correlations. To calculate these values, the input scene data need to be nominally spatially aligned. All results are stored in the database and used in other algorithms. For the OLI panchromatic band, all statistics are calculated and saved separately for odd and even frames. Processing of longer collects is described in Long Collect Statistics Characterization ADD.

7.4.4.2 Inputs

Descriptions	Symbol	Units	Level	Source	Type
Scene data at any processing level, including blind detectors and video reference pixels (VRP)	Q		N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Int or Float
Impulse Noise Locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Dropped Frame Locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Saturated Pixel Locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Detector offsets		Pixels	N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Int
Number of frames skipped at the top		Pixels	1	CPF	Int

Number of frames left out at the bottom		Pixels	1	CPF	Int

7.4.4.3 Outputs

Descriptions	Symbol	Units	Level	Target	Type
Minimum	Q_{min}	DN	N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Float
Maximum	Q_{max}	DN	N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Float
Mean	$\overline{Q}$	DN	N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Float
Standard deviation	σ	DN	N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Float
Skewness	γ_1		N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Float
Kurtosis	γ_2		N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Float
Number of valid frames	N_{valid_pixels}	Pixels	N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Int
Mean squared response	$\overline{Q^2}$	DN^2	N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Float
Adjacent detector correlation	ρ		N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Float
Position in processing flow (RPS level)			1	Db	String
Linearization LUT version			1	Db	String

Note: For the OLI panchromatic band, all output values need to be generated and saved separately for odd and even minor frames.

7.4.4.4 Options

Typically these data will be stored in the characterization database. For stand-alone processing, all these data may be output to a summary report with a header containing start date and time of acquisition, end date and time of acquisition, processing date and time, calculated frame rate, filename and entity ID. A report generation should be selectable in work order.

7.4.4.5 Procedure

For each active detector, *d*, including inoperable detectors, blind detectors and VRPs, for all bands and SCAs, except for the OLI panchromatic band:

1. Obtain number of valid pixels, $N_{valid\_pixels}\ (d)$, in image or collect. Pixels identified in the label mask as saturated, impulse noise affected, or parts of dropped frames, as well as filled pixels used to generate detector offsets and pixels corresponding to the Number of frames to be skipped at the top and bottom of the image, are considered invalid and need to be taken out of calculations. The symbol *l* is used to denote the list of valid pixels.

2. Find the minimum of valid pixel values, *l*:

$$Q_{\min}(d) = \min\left(Q(l,d)\right)$$

3. Find the maximum of the valid pixel values, *l*:

$$Q_{\max}(d) = \max\left(Q(l,d)\right)$$

4. Calculate mean as:

$$\overline{Q}(d) = \frac{1}{N_{valid\_pixels}(d)} \sum_{l=1}^{N_{valid\_pixels}} Q(l,d)$$

5. Calculate standard deviation as:

$$\sigma(d) = \sqrt{\frac{1}{N_{valid\_pixels}(d)} \sum_{l=1}^{N_{valid\_pixels}} (Q(l,d) - \overline{Q}(d))^2}$$

6. Calculate skewness as:

$$\gamma_1(d) = \frac{\sum_{l=1}^{N_{valid\_pixels}} (Q(l,d) - \overline{Q}(d))^3}{N_{valid\_pixels}(d) \times \sigma(d)^3}$$

   If $\sigma(d) = 0$, set $\gamma_1 = 0$.

7. Calculate kurtosis as:

$$\gamma_2(d) = \frac{\sum_{l=1}^{N_{valid\_pixels}} (Q(l,d) - \overline{Q}(d))^4}{N_{valid\_pixels}(d) \times \sigma(d)^4} - 3$$

   If $\sigma(d) = 0$, set $\gamma_2 = 99999$.

8. Calculate correlation between each detector (image column) and the neighbor on its right side using only pixel pairs where adjacent pixels from both detectors are valid. This step needs to be performed for all detectors except the last one on each SCA.

$$\rho_{d,d+1}(d) = \frac{1}{N_{valid\_both}} \sum_{l=1}^{N_{valid\_both}} Q(l,d)Q(l,d+1)$$

where N_{valid_both} is the number of valid adjacent pixel pairs. A pixel pair is valid if both adjacent pixels in spatially aligned image, generated by two neighboring detectors, are valid.

9. Calculate mean squared response:

$$\overline{Q^2}(d) = \frac{1}{N_{valid\_both}} \sum_{l=1}^{N_{valid\_both}} Q(l,d)^2$$

10. Save results to the database and report file (if that option selected in work order)

For each active detector, *d*, in the OLI panchromatic band:

Repeat the steps 1 to 10, but separately for odd and even frames.

7.4.5 Coherent Noise Characterization

7.4.5.1 Background/Introduction

Coherent noise (CN) is typically picked up somewhere in the analog data stream from the detectors to the A/D convertors. Such noise generates a temporally correlated structure, therefore it is most easily detected in uniform images. Both TIRS and OLI have a coherent noise requirement. Trending of coherent noise parameters from this algorithm will help in monitoring system health and document lifetime changes of the LDCM payloads. The heritage coherent noise characterization algorithm was successfully implemented in the ALIAS system.

The algorithm extracts the CN components (as defined in the system level requirements) and stores the coherent noise parameters in the trending database. Once sufficient data have been processed using this algorithm and the results stored in the database, additional analysis operations will be used to generate lists of detectors that have CN components that are near or exceed system requirements (OLI-985 and TIRS-532).

Compliance with the CN requirement will be assessed using low dynamic range scenes, for example: OLI shutter, OLI night Earth/ocean , OLI solar diffuser, TIRS deep space, or TIRS on-board BB. Any acquisition can be used for the OLI and TIRS blind bands .

The prototype code submitted with this ADD implements the simplest case of 2 scene types: OLI-Dark shutter data and TIRS-Cold Space.

Input Data Prerequisites:
To ensure a low noise floor and low uncertainty a minimum of 6400 lines for OLI and 4200 lines for TIRS are required. An analyst should be able to run this algorithm on any length of data as long as the output is not stored into the operational trending database.

The algorithm also allows averaging Power Spectral Densities (PSD) from up to 10 different data collects (consistent with the system requirements) of the same size (10 per payload may mean up to 20 files total). The number of input files used needs to match the algorithm “Multi_Scene_Counter” input parameter. By default this parameter will be “1” , which pre-launch analysis shows is adequate and could be as high as 10 (this limitation to the input parameter valid range will occur only when the trending option is selected).

Note: if trending is turned off, any number above 1 is valid for the “Multi_Scene_counter”, i.e., the limit of 1-10 only applies to operational trended data analysis and processing.

The CN processing and analysis will be done for each FPM for OLI and for the entire FPA (all bands all SCAs) for TIRS.

CN parameters extracted for trending (for major CN components):
(typically only for failing or near failing detectors):
- Relative Amplitude (%),
- Frequency (cycles/pixel),
- Modulation (DN)
- Uncertainty statistics.

Overview characteristics for trending:

Master frequency list histogram information – all detectors all bands (all FPMs)

Pass/Fail/uncertain mode determination (implemented internally in code or by post processing DB query):

Review plots:
For OLI -one per Band per FPM of Relative Amplitude vs. Frequency overlaid with the requirement thresholds that have been adjusted by noise floor baseline.
For TIRS –One for the worst detectors in the entire FPA for both bands of Relative Amplitude vs. Frequency overlaid with the requirement thresholds that have been adjusted by noise floor baseline.

Dependencies

The input data used are dependent on the trending flag setting. For operational trending the input data should be processed by IAS RPS up to but not including gain application, while for non-trended data the analysis code could also be run on L0ra input data (when the mean data level will be subtracted as part of the analysis).
Depending on the payload data source, the operational trending processing could include only bias removal, or both non-linearity correction and bias removal. If multiple scenes are used they should all be for the same object and the same sensor.

7.4.5.2 Inputs

Descriptions	Symbol	Units	Level	Source
Payload image data (partially processed)	$Q_{B,S,D,L}$	Float [DN]	Multi_Scene_Counter x[N_BxN_SxN_DxN_L]	In-line processing outputs
Number of input scenes	Multi_Scene_Counter	Integer [Unitless]		ADD running parameter GUI
Input scenes ID or WO ID reference table	Ref_files	String array	[Multi_Scene_Counter x 2]	Excel input file
Sensor (i.e OLI or TIRS)	Stype	Char	[Multi_Scene_Counter x 2]	?(from data file name /input data header info)
Scene object	Sobj	Integer	[Multi_Scene_Counter x 2]	Flag indicating scene

				object (i.e. Dark shutter, deep space
Impulse Noise Pixels Locations	LM1	Integer	Multi_Scene_Counter x N_BxN_SxN_D xL	LM
Saturated Pixel Mask[6]	LM2	Integer	Multi_Scene_Counter x N_BxN_SxN_D	LM
Dropped Frames Mask	LM3	Integer	Multi_Scene_Counter x N_BxN_SxN_D xL	LM
Inoperable Detector List	Dinop	Float	N_BxN_S	CPF
SNR scaling factors	SNR_scale	Float	N_b	Default is 1 For all bands Analyst input (from SER Doc Reference)
Analysis Length[7]	Ana_leng	Integer	[2]	Defaults is [6400,4200] post OIV Analyst can select any other length manually
Trending Flag	TF	Boolean	On/off	Analyst input
Max number of stored CN candidate components	Freq_bins	Integer	[2]	Default is [610,400] , if Trending Flag OFF could be set by analyst.

7.4.5.3 Outputs

Descriptions	Symbol	Units	Level	Target

[6] This mask should include any of the detector samples that were reported by the Saturation characterization processing, i.e. high and low saturations.

[7] When trending flag option is ON if the desired analysis length is not met algorithm will not run and report the user about input data length error. By default the scene will be clipped to the number of lines defined by this parameter.

CN Relative Amplitude	Amp%	Float [%]	N_B x N_S x N_D x Freq_bins	DB
CN component Frequency	Freq	Float [cycles/pixel]	N_B x N_S x N_D x Freq_bins	DB
CN state flag	uncertain /Fail	Fail – uncertain	N_B x N_S x N_D	DB
Noise floor	NF	Float [DN^2/ cycle/pixels]	N_B x N_S x N_D	DB
Scene Type (OLI or TIRS)	Stype	Char	2	DB
Analysis length (if trend option is not selected)	Ana_len	Integer	Scene Type (OLI or TIRS)	DB
SNR scaling (if option was selected)	Scale_snr	Float []	N_B x N_S	DB
Scene object	Sobj	Integer	2	DB
Reference file WO or IDs	Ref_files	String array	Multi_Scene_Counter	DB
CN component DN modulation and related statistics	CN_amp_dn	Float [dn]	Possible to store only Top level CN components rather than the whole FPA data Max size will be N_B x N_S x N_D	DB
Master Freq list 1 per detector per band (For OLI)	Freq_list1	Float	[N_D x N_B x freq_bins]	DB
Master Freq list 2 per detector per FPA (for TIRS)	Freq_list2	Float	[freq_bins]	DB
Processing Step	Pstep	Integer	[2]	DB

Note: the effective number of “Freq_bin” is determined by the analysis and it can be different for different detectors within a band, or for different bands the reminder array will be filled with null values. The variable Freq_bin is maximum number of CN components that need to be evaluated against the requirements. For TIRS data this value is limited to 400 for OLI it is limited to 610. The theoretical maximum is half of the Analysis Length. Typically, up to 30 unique frequencies bins will result.

Other outputs that could be stored in analysis archive or be included in a report are:
Plots of CN Vs Freq per band per FPM/SCA (for imaging detectors)
Lists of Frequencies and Amp % of top CNs per band FPM/SCA or entire FPA
CN state flags (highlighting detectors that are in Fail or uncertain requirement)

7.4.5.4 Options

Trending On/Off Switch: If trending is Off, output parameters are written to a dump text file, and length of data pre-requisite is not checked, also the maximum number of input file will be unlimited and input source is not required to run through IAS processing hence no LM inputs are required. If Trending flag is ON output data is to be trended in official operational IAS DB, data will be clipped to

the default analysis length, input data should have the accompanying LM and it should be pre-processed by the IAS RPS to the required level.
SNR Scaling: Use SNR scaling on/off (default is off i.e. scaling =1) when SNR scaling is On scaling parameters will be provided by the analyst as additional input parameter variable (one scale value per band - as the smallest ratio between all FPM SNRs in that band and requirement SNR – this ratio will be calculated from known SNR analysis data in a separate manual analysis)

7.4.5.5 Procedure

1) Determine options selected for SNR scaling and trending

Per Payload data type repeat the sequence from 2-7

CN analysis preparation steps (2-4)

2) Get list of input files process them as required (through IAS RPS without gain application if trending is selected). If the trending flag option selected is “no-trending” input files could be also the raw-L0ra data and not include LM inputs.

3) Get input parameters as indicated in the table above – depending on trending flag option and SNR scaling option the input parameters list maybe different

Note that if multiple files are used as input the trending analysis could only be limited for detectors that are not included in any of the aggregated (“OR” logic operation) Label Mask lists.

4) If trending to operational DB is selected check that the requirement of minimum data lines are met based on input data type. Cut the data size to the pre-selected default analysis length. If the minimum requirements are not met signal an error to the reviewing analyst. (if no trending selected analysis length is equal to shortest data length of the input files)

Core CN processing steps

5) For each FPM in OLI or SCA in TIRS:
5.1 For each Band/Detector:

5.1.1 Ignore detectors that are inoperable (based on CPF) or are included in the aggregated multi-scene outlier list i.e. have any saturated samples, impulse noise or dropped frames as given in the LM if all scenes under consideration.

5.1.2 Generate a Power Spectral Density (PSD) for each detector from each data file.

5.1.3 Average the results of all PSDs for each detector and filter the average-PSD

5.1.4 Extract the noise-floor baseline from the average-PSD.
Current implement approach uses a 3 step semi-iterative filtering operation
First the processing constructs a filter kernel using 2 parameters of the filter cut-off and the filter cut-off slope. In the current algorithm this is done by a custom filter in the mathematical form shown here:

$$Filter(t) = \frac{1.0}{1 + (\frac{tri\_func}{data\_len \times A})^{B}} \qquad \text{(Eq 1)}$$

Where the parameters used are A= 0.0195312 (slightly over 1/25 of Nyqiust)
And B=10

Next the 3 step filtering occur as follows by the diagram in Figure 1 and the following description
Step 1- use the constructed filter kernel to low pass filter the PSD
Step 2- smooth the result further with boxcar filtering a.k.a. convolving with Rect function with a width C where C=data_length/60
Step3 – compute noise floor by passing the results once more through the constructed filter kernel.

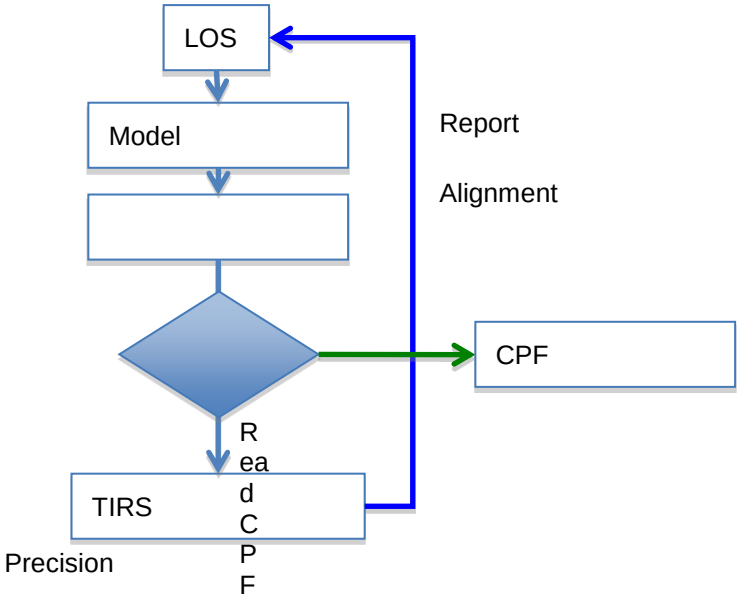


Figure 1. Noise floor extraction from PSD

5.1.5 Determine candidate frequency bins for CN evaluation. Based on statistical deviation of the average PSD from the noise floor.
This is done in current code as follows:
Compute the refined standard deviation and mean od the ratio of the noise_floor corrected PSD to the noise_floor.
The ratio is defined here as Ratio1

$$Ratio1_{s,b,d} = \frac{Absolute\_value(PSD - noise\_floor)}{noise\_floor} \quad \text{Eq (2)}$$

The refined statistic Mean2 and StDev2 are computed after the data the cropped to the range of +/-5 sigma from the mean of the initial standard deviation and mean.
Then the refined statistics on the Ratio1 is used along with the noise floor to construct a threshold level to the PSD for detection of PSD spikes that are candidate CN components.
This ratio is defined here:

$$Threshold0_{s,b,d} = (mean2 + 5 \times StDev2) \times Noise\_floor \quad \text{Eq (3)}$$

All PSD frequencies where Ratio1 is above Threshold0 are considered candidate CN components.

5.1.6 Calculate the uncertainty (1 sigma) in PSD in both

DN^2/Hz units and %/Hz units.
In current code this is computed as described by the two equations.

$$Sigma3_{s,b,d} = StDev2 \times Noise\_floor \quad \text{Eq (4)}$$

$$Sigma4_{s,b,d} = \sqrt{2} \times \sqrt{Sigma3}_{s,b,d} / (3 \times Scene\_stdev_b) / \text{Spec\_SNR\_ratio} \times 100[\%] \quad \text{Eq (5)}$$

5.1.7 If trending flag selected the operational trending, confirm that the analysis uncertainty is good enough, based on the analysis uncertainty for the PSD at the Nyquist frequency. The criteria is set to be 0.842*Sigma4 (at Nyquist)< 0.5%. If this is not met, then more data collects averaging are needed. Halt processing for analyst review.

5.1.8 Test the candidate frequency bins for compliance with CN requirement. This requires a conversion of the average PSD to %Amp. Compute and Store %Amp and frequencies output.

This is done be defining a threshold1 level

$$Threshold1_{s,b,d} = 0.842 \times Sigma4_{s,b,d} \quad \text{Eq (6)}$$

This threshold defined the confidence level of 80%
For testing the requirement the CN components are tested against the requirement+Threshold1
Where CN > requiremt+threshold1 : fail
CN < requiremt-threshold1 : pass
CN between those two conditions then it is in the uncertain-state category – only trending may determine if it is pass or failed condition.

5.2 Store summary frequency histograms for all candidate CN components (even if they pass the requirement)
5.3 Store the reminder of output parameters

Evaluation and Analysis steps

6) Generate plot of CN per Band per FPM/SCA or per FPA
7) Generate Band or FPA level summary report.

The flow of the processing is illustrated in the following diagram:

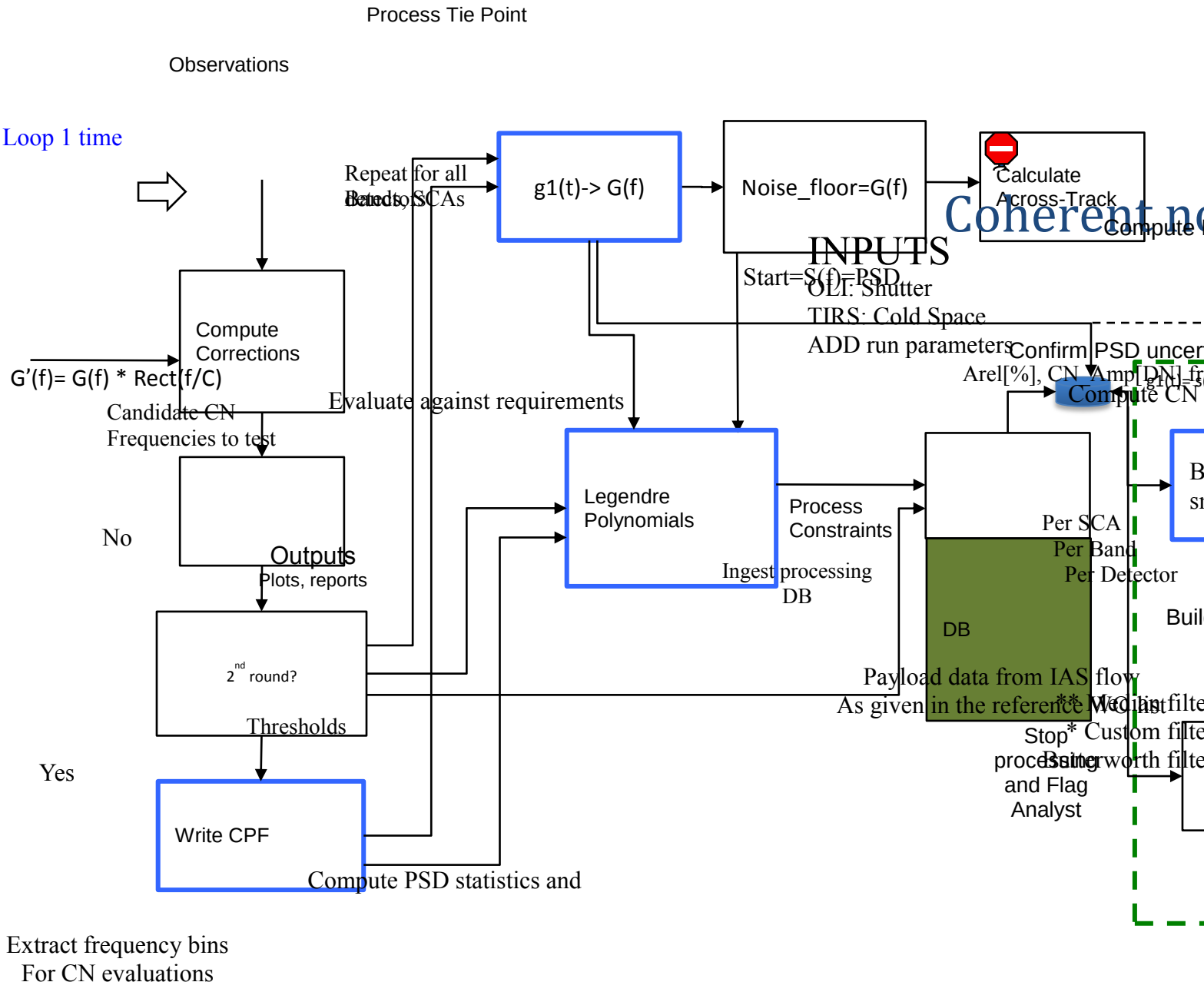


Figure 2. CN Toolkit processing flow.
(ADD steps are indicated on the diagram by the red numbers.)

7.4.5.6 Maturity

Level 1 (reuse and update). No major changes anticipated to ADD however various revision to code could be implemented for enhancements and accommodation of new input data objects, or extraction of more characterization parameters for the non-trending debug and algorithm maintenance mode.

7.4.6 Temperature Sensitivity Characterization

7.4.6.1 Background

Focal plane temperature variations affecting gain and offsets, were considered for correction during Landsat and EO-1 ALI processing. A similar correction for gain and bias will be available for both OLI, and TIRS during LPGS data processing.

The acquisitions using the on-board cal sources, and their associated focal plane thermistor data, are used to determine and characterize the temperature sensitivity coefficients (C_T) used in correction. For OLI, the (primary) diffuser and (primary) stim lamp collects, and average of 2 focal plane thermistor readings will be used. For TIRS, the OBC collects and the average of 4 focal plane thermistor readings will be used. The regression slopes of response to temp will provide the coefficients.

Note, while the Focal Plane temperatures are identified, it's conceivable that Focal Plane Electronics temps could also be used.

Note, while all thermistors are assumed to operate nominally at launch, post-launch, these thermistor will be continuously characterized i.e. temperatures will be trended and evaluated per interval and over the mission life ("globally") Should this characterization reveal outliers and/or any thermistors falling "out of family", that thermistor will be evaluated for exclusion. It's assumed the associated thermistor readings have been converted to degs C and any time offset in the temperature telemetry has been corrected prior to extraction into this tool.

7.4.6.2 Input

Descriptions	Symbol	Units	Level	Source	Type
Focal Plane Temperatures (OLI)	FPM_7_TEMP_CELSIUS (FPM 7 Temp)= T_1 FPM_14_TEMP_CELSIUS (FPM 14 Interface Temp) = T_2	C		Db	Float
Focal Plane Temperatures (TIRS)	FP_F2_FINE_SENSOR_1_CELSIUS =T1 FP_F4_FINE_SENSOR_3_CELSIUS =T2 FP_F6_FINE_SENSOR_1_CELSIUS =T3 FP_F7_FINE_SENSOR_2_CELSIUS =T4	C		Db	Float

OLI Diffuser or Lamp (PRIMARY) Gains	$G_{OLI\text{-}S}$ or $G_{OLI\text{-}L}$	DN/ W/m^2-mu-sr	N_{band} x N_{det}	Db	Float
TIRS OBC Gains	G_{TIRS}	DN/ W/m^2-mu-sr	N_{band} x N_{det}	Db	Float
Start Time, and Stop Times	T_s, T_f	Secs			

7.4.6.3 Output

Descriptions	Symbol	Units	Level	Source	Type
OLI Temp (Means and Stdevs)	$<T_n>$, σT_{n-} where n=1,2	C			Float
OLI Mean Temp	<T>	C			Float
OLI Temp Sensitivity Coeff and uncertainty	C_{T-H} ΔC_{T-H}	DN/C	$N_{Band}xN_{SCA}xN_{Det}$	CPF	Float
TIRS Temp (Means and Stdevs)	$<T_n>$ $\sigma T_{n-Stdev}$ where n=1,2,3,4	C			Float
TIRS Mean Temp	<T>	C			Float
TIRS Temp Sensitivity Coeff and uncertainty	C_{T-O} ΔC_{T-O}	DN/C	$N_{Band}xN_{SCA}xN_{Det}$	CPF	Float

7.4.6.4 Options

OLI: Lamp or Diffuser Data Input; Time Range of Data,
TIRS: OBC temp; Time Range of Data

7.4.6.5 Procedure

Preparation: In advance of running this algorithm the variation in the various focal plane related temperatures should be trended. If variation is observed, then similar variations in these temperatures should be looked for in calibration acquisitions (lamp and solar for OLI and OBC at a fixed BB temp for TIRS). If variation is observed in the calibration acquisitions, the appropriate temperatures and instrument responses should be extracted from the IAS database.

For each calibration interval selected

For each thermistor (T_n)

1) Generate temperature avg's ($<T_n>$) and stdev's, (σT_n)

2) Reject any "outlier" thermistor and set thermistor flags to 0 in CPF.

End ; thermistor loop

3) Average across all valid thermistors for the interval (<T>)

End ; cal interval

4) For a selected instrument calibration type, over a give range of start (T_s) and stop times (T_f) , extract and regress the instrument gains ($G_{OLI-S/L}$ or G_{TIRS}) to the avg focal plane temperature (<T>)

5) Calculate the temperature sensitivity coefficient C_T (where C_T is inverse of correction factor per Jackson and Robinson, (1985),) by ratioing the slope of the regression to the intercept, when the regression is significant. A value of '0' translates to no temperature correction.

7.4.7 Temperature Sensitivity Correction

7.4.7.1 Background

Focal plane temperature variations affecting gain and offsets, were considered for correction during Landsat and EO-1 ALI processing. A similar correction for gain and bias will be available for both OLI, and TIRS during LPGS data processing. There are 2 thermistors on the OLI focal plane. The TIRS focal plane has 8 thermistors, only 4 of which will be provided as part of the ancillary data.

This algorithm uses an average of each set of focal plane temperatures to derive the temperature correction. This is a linear correction and is done on a per-detector basis. These thermistors are available in the ancillary data and are updated every 1 second for OLI. (This is anticipated to be the same for TIRS) . This algorithm assumes that these thermistors readouts have been converted to K and any time offset in the temperature telemetry has been corrected prior to usage in this algorithm.

7.4.7.2 Input

Descriptions	Symbol	Units	Level	Source	Type
Temp Sensitivity Coefficient	C_T	-	$N_{Band}xN_{SCA}xN_{Det}$	CPF	Float
Scene Focal Plane Temperatures (OLI)	FPM_7_TEMP_CELSIUS (FPM 7 Temp)= T_1 FPM_14_TEMP_CELSIUS (FPM 14 Interface Temp) = T_2	C	Per Interval	Ancillary	Float
Scene Focal Plane Temperatures (TIRS)	FP_F2_FINE_SENSOR_1_CELSIUS (SCA-A Edge Sensor)=T1 FP_F4_FINE_SENSOR_3_CELSIUS (SCA-B Edge Sensor)=T2 FP_F6_FINE_SENSOR_1_CELSIUS (Center FPA Sensor) =T3 FP_F7_FINE_SENSOR_2_CELSIUS (Center FPA Edge Sensor) =T4	C	Per Interval	Ancillary	Float
Reference Temperature	T_{Ref}	K	Per Instrument	CPF	Float
Thermistor_Flag	T_{Flag}	-	Per Instrument Thermistor i.e. 2 OLI, 4 TIRS	CPF	Int

7.4.7.3 Output

Descriptions	Symbol	Units	Level	Target	Type
Correction Factor	CF_T	-	N_{Band} xN_{SCA}xN_{Det}	Gain Application	Float

7.4.7.4 Options

The invocation of this algorithm and the application of CF_T , can be performed at any stage prior applying gain during LPGS processing.
By default, this correction will be OFF during LPGS processing as specified via work order parameter.

7.4.7.5 Procedure

1) For each set of instrument thermistors #1 to #n, extract "scene-equivalent" temperatures based upon n-thermistor sample times and scene-start/stop times, where n=2 for OLI and n=4 for TIRS

$T_1 = T_1$ where $((t_1$ GE $ts_i)$ and $(t_1$ LE $ts_f))$
…
$T_n = T_n$ where $((t_n$ GE $ts_i)$ and $(t_n$ LE $ts_f))$

Where t_{Si} = Scene Time initial
t_{Sf} = Scene Time final
t_1 = Thermistor-#1 times
t_n = Thermistor-#n times

2) Average each Focal Plane temperatures (from Step 1) and derive single average Focal Plane temperature (T_{FP}) i.e.

$T_{1\,AVG}$ = Sum of all T_1 / Total # of T_1
…
$T_{n\,AVG}$ = Sum of all T_n / Total # of T_n

3) Multiply each single avg Focal Plane temp by its corresponding Thermistor Flag value i.e.

$T_{1AVG} = T_{1AVG}$ x T_{1Flag}
…
$T_{nAVG} = T_{nAVG}$ x T_{nFlag}

Where

T_{1Flag} = 1 (default)
…
T_{nFlag} = 1 (default)

Note, the value of each Thermistor flag is “good” (=1). Should a thermistor become “bad ”, its corresponding Thermistor flag will be set = 0.

4) Average all “good” thermistor avg values from Step 3) i.e.

$T_{FP} = (T_{1AVG} + \ldots + T_{nAVG})/n$

5) For each Band, SCA and detector, using the equation in Step 4) and other input parameters, to derive and output the Correction Factor (CF_T) i.e.

$CF_T = [(1 + C_T*T_{FP})/(1 + C_T*T_{Ref})]$

Where C_T= Temperature sensitivity coefficient
(This is the inverse coef used by Jackson etal, 1985)
T_{FP} = Focal Plane temperature (K)
T_{Ref} = Reference temperature (K)

Suggested implementation during PGS processing
1) Run algorithm prior to “Apply Gain” algorithm
2) Input TCorr to “Apply Gain” algorithm and multiply by the applied gain.

7.4.8 Gain Application

7.4.8.1 Background

Generation of Level 1r products result in an estimated in-band radiance product in W/m^2-sr-μm. This conversion occurs in 3 steps/algorithms; bias subtraction, non-linearity correction and gain (Absolute and Relative) application. Processing options allow the generation of intermediate calibrated products – e.g. with bias correction only, or non-linearity correction only, etc.

The Gain Application algorithm addresses the final step in generating the L1r radiance product. The type of gain application can be selected by parameters within the processing work order. Absolute gain application applies the same gain to all detectors within a band and SCA. Relative gain application involves absolute gain application, but then an additional gain is applied to each detector, to correct for slight differences between individual detector gains.

The last step in gain application allows for the optional processing of gain temperature sensitivity corrections. The Temp Sensitivity Correction Algorithm description describes the calculation of the sensitivity correction parameters.

7.4.8.2 Input

Description	Symbol	Units	Level	Source	Type
Scene (bias corrected, linearized)	*Q*	DN	N_{bands} x N_{SCAs} x $N_{detectors}$	Response Linearization	Float

			x N_{frames}		
Absolute Gain	G_{abs}	DN m^2 sr μm/W	N_{bands} x N_{SCAs}	CPF	Float
Relative Gain	G_{rel}	Unitless	N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Float
Temperature Sensitivity Coefficients	CF_T	Unitless	N_{bands} x N_{SCAs} x $N_{detectors}$	Temperature Sensitivity Correction	Float

Output

Description	Symbol	Units	Level	Target	Type
Scene (bias corrected, linearized, gain applied)	*L*	W/(m^2 sr μm)	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float

7.4.8.3 Options

- Apply absolute gain (default on)
- Apply relative gain (default on)
- Apply temperature sensitivity correction (default off)

7.4.8.4 Procedure

The first step in gain application is the decision of which method to use. Work order parameters should allow the operator to choose absolute gain application, relative gain application (which includes absolute gain application), or no gain application at all. Once those parameters are parsed, the respective gain parameters should be obtained from the CPF.

The absolute gain flag controls whether any gain is applied to the data. If the absolute gain flag is not set, no gain application is performed, although temperature sensitivity correction may still occur. This allows temperature sensitivity correction to be performed by itself, for testing.

If the absolute gain flag is set, the output value (*L*) is the linearized DN value (*Q*) for the band divided by the absolute gain parameter value (G_{abs}).

$$L(d) = Q(d) / G_{abs}$$

where $L(d)$ = Output radiance value for detector 'd'.
$Q(d)$ = Input DN value (linearized and bias corrected) for detector 'd'.
G_{abs} = The band absolute gain.

If relative gain application is also desired, the absolute gain value (G_{abs}) is multiplied by the detector-dependent relative gain parameter value (G_{rel}) prior to dividing into the linearized DN value (Q).

$$L(d) = Q(d) / [G_{abs} * G_{rel}(d)]$$

where $G_{rel}(d)$ = Relative gain parameter for detector 'd' of this band.

If the temperature sensitivity correction flag is set, temperature sensitivity correction is then performed. The form of this function is hardware-dependent, and may change when the characteristics of the instrument become known. The expected form of the temperature sensitivity correction is:

$$L^*(d) = L(d) * CF_T(d)$$

where $L^*(d)$ = Output temperature-corrected radiance value for detector 'd'.
$CF_T(d)$ = Temperature sensitivity correction coefficient for detector 'd'.

The temperature sensitivity coefficients are obtained from the temperature sensitivity characterization function, which is described in the Temperature Sensitivity Correction ADD. Note that the output of that characterization function does not currently use the housekeeping temperatures, but the final version is expected to require them.

7.4.9 L1R SCA Stitching

7.4.9.1 Background

During product generation the OLI and TIRS data are radiometrically processed on an SCA by SCA basis, but stitching of data from all SCAs to the instrument's full-field-of-view product for each band does not occur until Geometric Processing. Certain radiometric assessments at various stages in the processing flow, e.g. Non-uniformity Characterization, need to be performed on the entire band images.

This algorithm uses the SCA offsets to nominally spatially align all SCAs within each band and stitches the SCAs together to generate L1R single band images. The generated single band images can then be combined to form nominally aligned multispectral images. The algorithm assumes that the input data within each SCA are nominally spatially aligned, i.e. the odd/even detector offsets and the offsets for the selected non-primary detectors have been applied. Further, it is assumed that information about the number of imaging detectors per SCA (SCA width, in pixels) and the SCA length, in pixels, are available (Fig. 1).

The stitching can be accomplished by handling the SCA overlap regions in several ways:

- Method 1 – No Overlap; stitching without overlapping the SCAs (Fig. 1)
- Method 2 – Left Overlap; SCA1 is complete and all other SCAs miss the overlap region on their left side in the image (Fig. 2)
- Method 3 – Right Overlap; the last SCA is complete and all other SCAs miss the overlap region on their right side in the image (Fig. 3)
- Method 4 – Half-Half, SCA1 misses half of the overlap region on its right side, the last SCA misses half of the overlap region on its left side, and all other SCAs miss half of the overlap region on their both sides. While the geometric processing averages the overlap region, this method is the closest match.

7.4.9.2 Inputs

Descriptions	Symbol	Units	Level	Source	Type
Scene L1R data, except the blind band	L1Rp	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float
SCA overlap width	SCA_Overlap_Width	Pixels	N_{bands}	CPF	Int
Stitching method	Method		1	Work Order	Int or String
Detector offsets	O_{det}	Pixels	N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Float
Along Track Legendre Polynomial Coefficients			N_{bands} x N_{SCAs} x 4	CPF	Float
Inclination Angle	I	Degrees	1	CPF	Float
Nominal Orbit Radius	r_o	km	1	CPF	Float
Orbital Period	T_o	s	1	CPF	Float
Semi Major Axis	r_{eq}	m	1	CPF	Float
Semi Minor Axis	r_p	m	1	CPF	Float
Eccentricity	E		1	CPF	Float
Earth Angular Velocity	V_g	Radians/s	1	CPF	Float
Nominal Frame Time	t_s	ms	$N_{sensors}$	CPF	Float
Scene Center Latitude	ϕ	Degrees	1	Scene Metadata	Float
Nominal Scene Elevation*	d	km	1	DEM	Float
SCA Offsets	O_{SCA}	Pixels	N_{bands} x N_{SCAs}	CPF	Int

*The nominal scene elevation can be the average of the minimum and maximum elevation of the scene.

7.4.9.3 Outputs

Descriptions	Symbol	Units	Level	Target	Type
Nominally aligned L1R band image (L1Rp)	L1R	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float
Stitching method			1	metadata	Int or String

7.4.9.4 Options

The stitching method and list of bands to be stitched are selected through the work order. Additionally, the option to use just the SCA offsets from the CPF (O_{SCA}), without doing the extra calculations, enables non-Earth scenes to be aligned manually.

7.4.9.5 Procedure

6. For Earth scenes, calculate the SCA_Offset according to the algorithm described in the appendix. For non-Earth scenes, the O_{SCA} straight from the CPF is used.
7. Calculate the stitched product length as:

   L1Rp_Length = SCA_Length + max(SCA_Offset)

   The SCA length represents the worst case scenario; it is the length of an SCA after the maximum possible detector offsets are applied (SCA_Length = N_{frames} + max(round(O_{det}))).

8. For the sensor specific number of SCAs per band, NSCA, calculate width of the stitched product:
   a. If the Stitching method is “No Overlap”

      L1Rp_Width = NSCA x SCA_Width

   b. otherwise

      L1Rp_Width = NSCA x SCA_Width – (NSCA-1) x SCA_Overlap_Width

      Note that this equation assumes that all SCA overlaps have the same width. If that assumption proves inadequate, each SCA to SCA overlap width will need to be accounted for individually.

9. For each band:

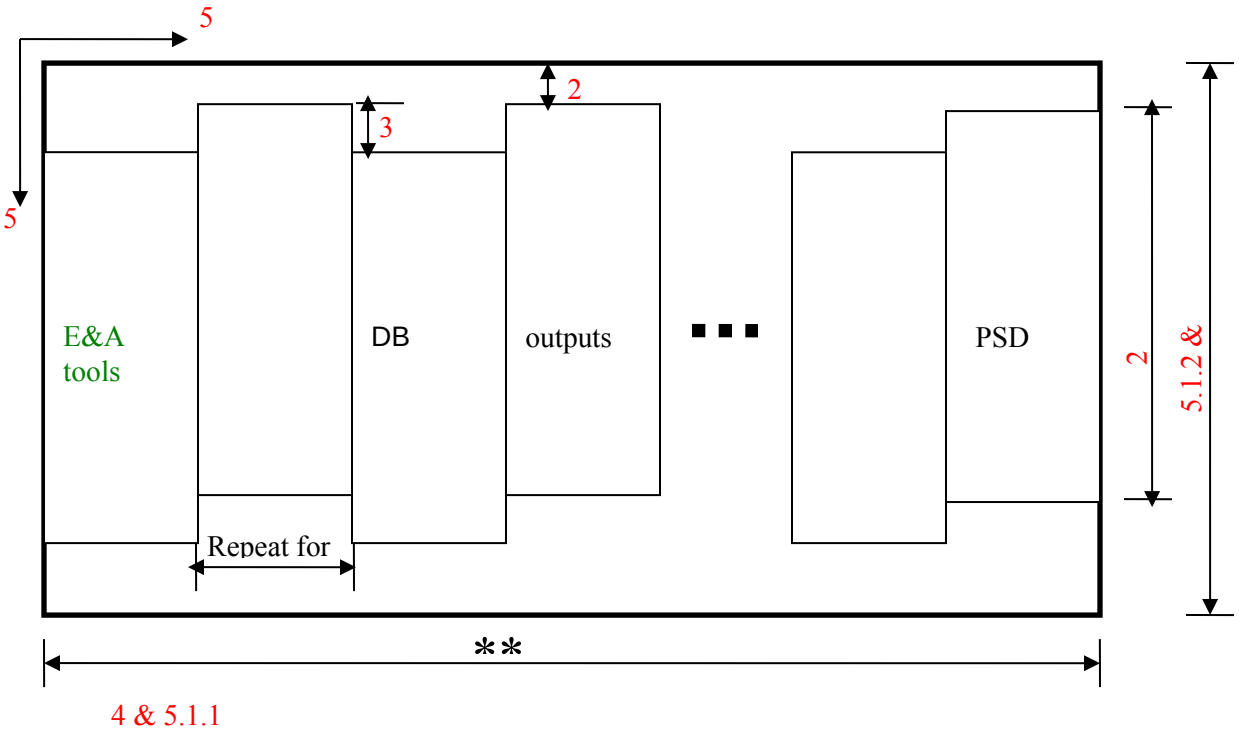
   a. Allocate an array, L1Rp, of size L1Rp_Length x L1Rp_Width and fill it with zeros.

   b. Load all SCAs

   c. If the Stitching method is “No Overlap” (see Fig. 1)
      i. For n = 1 to NSCA,

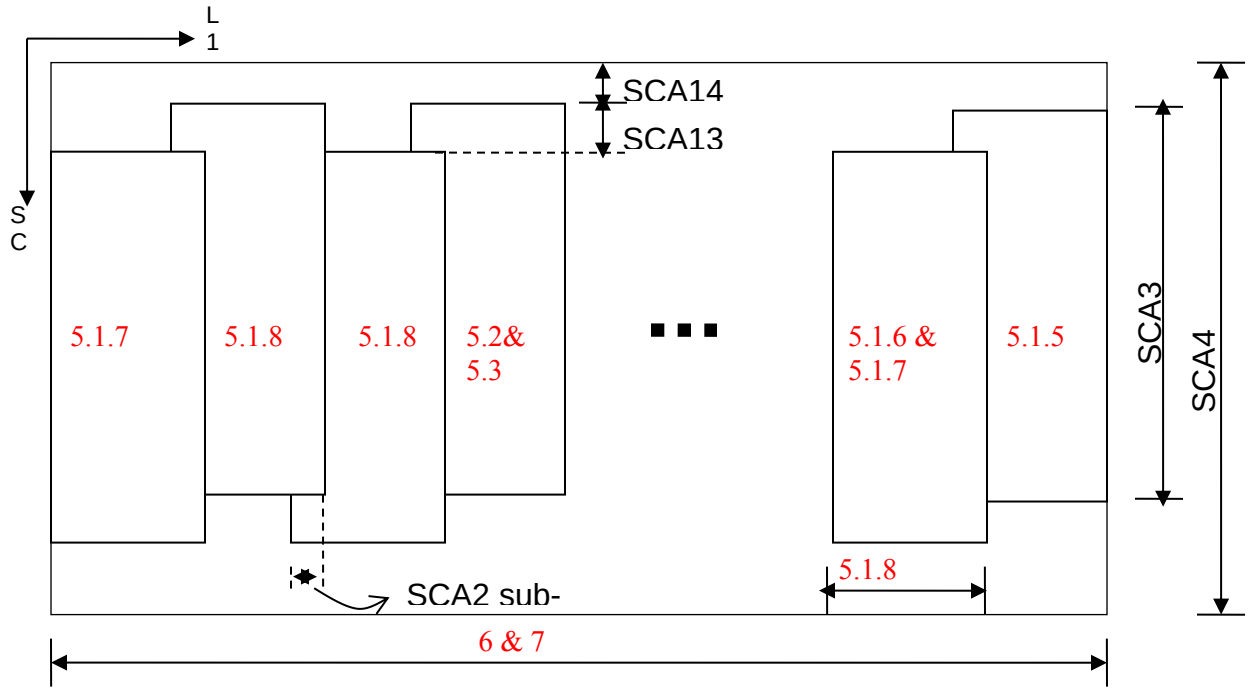
1. Copy the SCA(n) data to the sub-array of the L1Rp array defined in the 1-based coordinate system as:

	x (Column)	y (Row)
Upper Left Corner	(n-1) x SCA_Width + 1	SCA_Offset(n)* +1
Lower Right Corner	n x SCA_Width	SCA_Offset(n)* + SCA_Length



d. If the Stitching method is “Left Overlap” (see Fig. 2)
   i. For n = NSCA to 1 with step of -1,
      1. Copy the SCA(n) data into the sub-array of the L1Rp array defined in the 1-based coordinate system as:

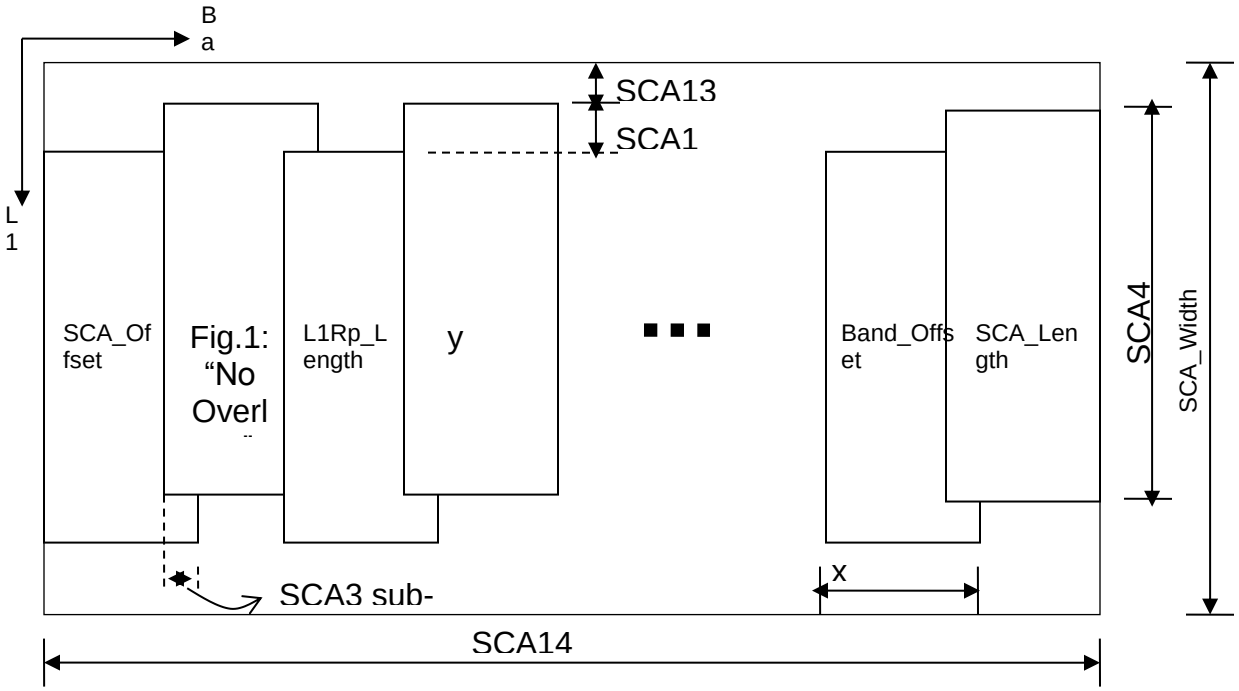
	x (Column)	y (Row)
Upper Left Corner	L1Rp_Width – (NSCA – n) x (SCA_Width – SCA_Overlap_Width) - SCA_Width + 1	SCA_Offset(n)* +1
Lower Right Corner	L1Rp_Width – (NSCA – n) x (SCA_Width – SCA_Overlap_Width)	SCA_Offset(n)* + SCA_Length



SCA1 sub-array

e. If the Stitching method is “Right Overlap” (see Fig. 3)
   i. For n = 1 to NSCA,
      1. Copy the SCA(n) data into the sub-array of the L1Rp array defined (in the 1-based coordinate system) as:

	x (Column)	y (Row)
Upper Left Corner	(n-1) x (SCA_Width - SCA_Overlap_Width) + 1	SCA_Offset(n)* +1
Lower Right Corner	n x (SCA_Width - SCA_Overlap_Width)	SCA_Offset(n)* + SCA_Length



SCA2 sub-array

f. If the Stitching method is "Half-Half" (see Fig. 4)
   i. Copy the SCA1 data into the sub-array of the L1Rp array defined in the 1-based coordinate system as:

	x (Column)	y (Row)
Upper Left Corner	1	SCA_Offset(1)* +1
Lower Right Corner	SCA_Width	SCA_Offset(1)* + SCA_Length

   ii. For n = 2 to NSCA,
      1. If SCA_Overlap_Width is an odd number
         a. Skip the first (SCA_Overlap_Width + 1) / 2 columns and copy the rest of the SCA(n) data into the sub-array of the L1Rp array defined in the 1-based coordinate system as:

	x (Column)	y (Row)
Upper Left Corner	(n-1) x (SCA_Width - SCA_Overlap_Width) + 1 + (SCA_Overlap_Width + 1) / 2	SCA_Offset(n)* +1
Lower Right Corner	(n-1) x (SCA_Width - SCA_Overlap_Width) + SCA_Width	SCA_Offset(n)* + SCA_Length

2. Else
    a. Skip the first SCA_Overlap_Width / 2 columns and copy the rest of the SCA(n) data into the sub-array of the L1Rp array defined in the 1-based coordinate system as:

	x (Column)	y (Row)
Upper Left Corner	(n-1) x (SCA_Width - SCA_Overlap_Width) + 1 + SCA_Overlap_Width/2	SCA_Offset(n)* +1
Lower Right Corner	(n-1) x (SCA_Width - SCA_Overlap_Width) + SCA_Width	SCA_Offset(n)* + SCA_Length

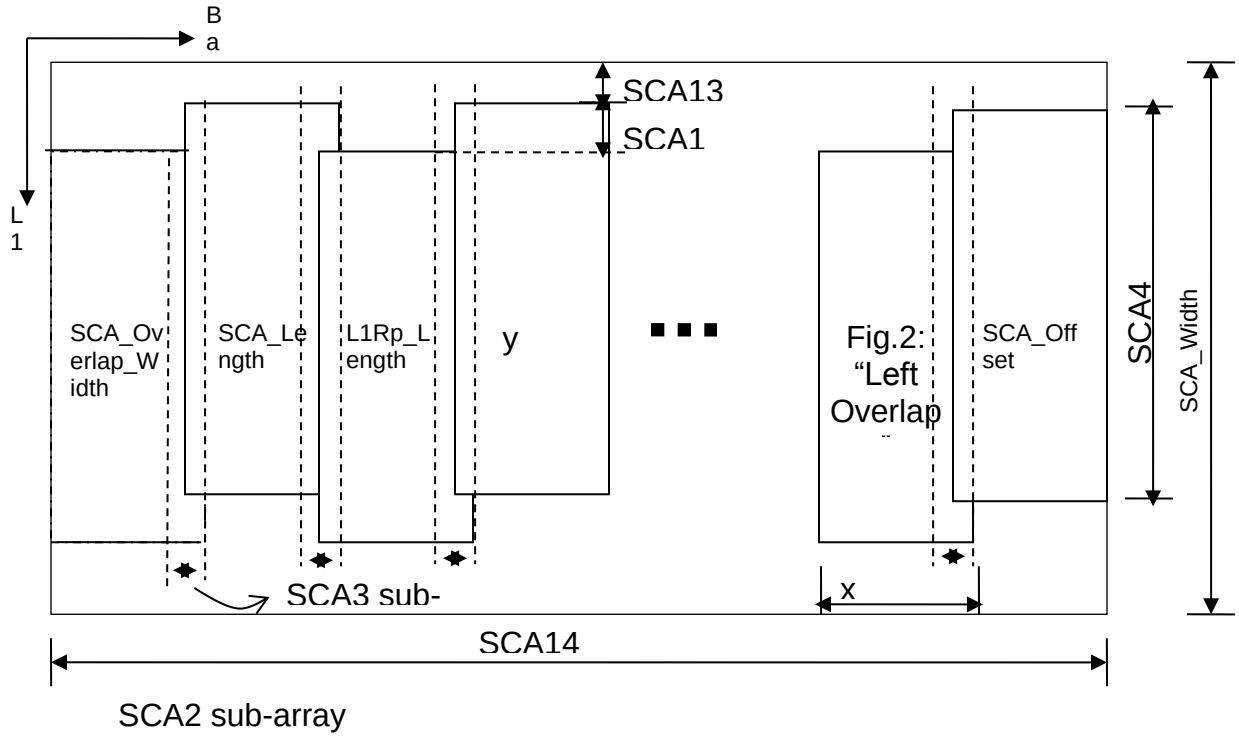


Fig.2: "Left Overlap"

10. Record the used Stitching method to metadata

7.4.10 Striping Characterization

7.4.10.1 Background

Evaluation of the effectiveness of relative gain correction to remove striping is typically performed in a qualitative sense, through visual inspection of imagery before and after correction. This method has several limitations, the primary one being that it relies on subjective human interpretation for the evaluation. In addition, inspection of large numbers of corrected images is not realistic. Consequently, a quantitative characterization of striping is needed.

Algorithms have been developed to quantitatively characterize striping through frequency-domain analyses (mostly FFT-based). These can produce an average estimate (across a focal plane module) of the amount of striping at the Nyquist frequency (corresponding to detector-to-detector variation). The disadvantage of this approach is that the results provide no real information about striping at a detector level (i.e. which detectors are more sensitive to striping, etc).

This algorithm determines a quantitative metric for the amount of striping present in an image through calculation of spatial-domain statistics for each detector. These statistics are further processed to obtain a "final" striping metric. In the initial development work it has been found that larger values for this metric tend to positively correlate with more visually apparent striping.

This algorithm will also produce a Striping Correction Matrix that may be used by the Residual Striping Correction algorithm to reduce that amount of striping in the image.

7.4.10.2 Inputs

Description	Symbol	Units	Level	Source	Type
Scene	Q	DN or W/m^2 sr μm	N_{bands} x N_{SCA} x $N_{detectors}$ x N_{frames}		Float
Saturated pixels			N_{bands} x N_{SCA} x $N_{detectors}$ x N_{frames}	LM	Int
Impulse noise			N_{bands} x N_{SCA} x $N_{detectors}$ x N_{frames}	LM	Int
Dropped Frames			N_{bands} x N_{SCA} x $N_{detectors}$ x N_{frames}	LM	Int
Inoperable detectors			N_{bands} x N_{SCA} x $N_{detectors}$	CPF	Int
Striping metric cutoffs			Nband	CPF	Float

The Striping Metric Cutoffs have a default of 2% of the standard deviation of "all" the images in the archive. This will be fairly constant, so there is not a need to query the database and calculate it every time. A value in the CPF should function adequately. Initialization of this parameter may be done after 100 images are in the archive/database.

7.4.10.3 Outputs

Description	Symbol	Units	Level	Destination	Type
Overall Striping Metric		DN or W/m^2 sr µm	Nband	Db	Float
Scene Striping Correction Matrix (optional)			Nband, Nsca, Ndet, Nframe	Residual Striping Correction Algorithm	Float
Detector Striping Metric (optional)			Nband, N_{SCA}	Report	Float
Scene Striping Metric (optional)			Nband, N_{SCA}	Report	Float

The Overall Striping Metric is a single number measure of the amount of striping found in the image.
The Scene Striping Correction Matrix is a matrix roughly the same size as the image. It can be subtracted from the image to remove striping.
The Detector Striping Metric is an $N_{det\text{-}2}$ array measure of the amount of the striping in each individual detector.
The Scene Striping Metric is an $N_{det\text{-}2} \times N_{frames\text{-}fill\text{-}2}$ measure of the amount of striping at each individual pixel.

7.4.10.4 Options

Write the Overall Striping Metric to database (On by Default)
If the Residual Striping Correction Algorithm is being run the Scene Striping Correction Matrix needs to be calculated and outputted. If the Residual Striping Correction Algorithm is not being run, the Scene Striping Correction Matrix does not need to be calculated or outputted.
Summary Report (Off by Default)

a) Detector striping metric
b) Scene striping metric
c) Overall striping metric

7.4.10.5 Procedure

1. Read in the processing parameters.
2. Read in an SCA.
3. Find the difference between every pixel in the image and the average of its two neighbors (left and right). When a pixel in the artifact mask or a sample from an inoperable detector is encountered, the Cross-Track Difference (CTDiff should be set to zero. So a pixel in the artifact mask or a sample from an inoperable detector will cause three entries to be zeros in CTDiff (the pixel itself and its two neighbors).

$$\mathrm{CTDiff} = x_{m,n} - \frac{x_{m,n-1} + x_{m,n+1}}{2}$$

Where *x* denotes a pixel and *m* and *n* denote row (frame) and column (detector) respectively. This difference is calculated for all pixels in the image except border pixels (one pixel on all sides).

4. Since scene content will cause the largest magnitudes in CTDiff, we will calculate a homogeneity filter.
   a. The first step is to check the Cross-Track Homogeneity (CTHom) by calculating the difference between pixels in the image on either side of the current pixel.

   $$\mathrm{CTHom(m,n)} = \left| x_{m,n-1} - x_{m,n+1} \right|$$

   This difference is calculated for all pixels in the image except border pixels (one pixel on all sides). Whenever a pixel in the artifact mask or a sample from an inoperable detector is encountered, CTHom should be zero, so all pixels in the artifact mask and their left and right adjacent pixels will have zero values in CTHom. Thus, inoperable detectors will cause three columns of zeros in CTHom.
   b. Next we will check the Along-Track Homogeneity (ATHom) by taking the vertical difference between the current pixel and its top and bottom neighbors.

   $$\mathrm{ATHom(m,n)} = \left| x_{m,n} - \frac{x_{m-1,n} + x_{m+1,n}}{2} \right|$$

   This difference is calculated for all pixels in the image except border pixels (one pixel on all sides). Whenever a pixel in the artifact mask or a sample from an inoperable detector is encountered, ATHom should be zero, so all pixels in the artifact mask and their top and bottom adjacent pixels will have zero values in ATHom. Thus, inoperable detectors will cause one column of zeros in ATHom.
   c. To help reduce noise, we will average CTHom over five pixels.

   $$\mathrm{ACTHom(m,n)} = \frac{1}{5}\left(\begin{array}{l}\mathrm{CTHom(m-2,n)} + \mathrm{CTHom(m-1,n)} + \\ \mathrm{CTHom(m,n)} + \mathrm{CTHom(m+1,n)} + \mathrm{CTHom(m+2,n)}\end{array}\right)$$

   ACTHom stands for Average Cross-Track Homogenity. This is done for the entire CTHom image. Pixels in the artifact mask or a sample from an inoperable detector and their left and right adjacent pixels should not be used to calculate this average. Border pixels are averaged with their inside neighbors, this can be seen below for pixels on the left side. Pixels on the right side use similar equations.

   $$\mathrm{ACTHom(1,n)} = \frac{\mathrm{CTHom(m,n)} + \mathrm{CTHom(m+1,n)} + \mathrm{CTHom(m+2,n)}}{3}$$

   $$\mathrm{ACTHom(2,n)} = \frac{\mathrm{CTHom(m-1,n)} + \mathrm{CTHom(m,n)} + \mathrm{CTHom(m+1,n)} + \mathrm{CTHom(m+2,n)}}{4}$$

   d. To reduce noise in ATHom, we will average over three pixels.

   $$\mathrm{AATHom(m,n)} = \frac{\mathrm{ATHom(m,n-1)} + \mathrm{ATHom(m,n)} + \mathrm{ATHom(m,n+1)}}{3}$$

   AATHom stands for Average Aross-Track Homogenity. This is done for the entire ATHom image. Pixels in the artifact mask or a sample from an inoperable detector and their top and bottom adjacent pixels should not be used to calculate this average. Border pixels are averaged with their inside neighbors, this can be seen below for pixels along the top border. Pixels along the bottom border use a similar equation.

   $$\mathrm{AATHom(m,1)} = \frac{\mathrm{ATHom(m,n)} + \mathrm{ATHom(m,n+1)}}{2}$$

e. To complete the homogeneity filter, plug ACTHom and AATHom into the equation below.

$$\mathrm{HomFilt}(m,n) = \frac{1}{1+\left(\frac{\mathrm{abs}\left(\mathrm{ACTHom}(m,n)+\mathrm{AATHom}(m,n)\right)}{\mathrm{Striping\ Metric\ Cutoff}}\right)^4}$$

HomFilt stands for Homogeneity Filter. This will generate a filter mask of roughly 1s and 0s the same size as the original image minus one border pixel from all sides. All pixels in the artifact mask and samples from inoperable detectors and left and right adjacent pixels should be zeroed out. This filter should remove scene content from the calculation of the striping metric.

5. The scene correction matrix is the individual pixel product of the HomFilt and CTDiff. The scene striping metric is essentially the absolute value of the scene correction matrix, but without the divide by two. The scene striping metric shows where, spatially, in the image stripes are located. The higher the value the more striping present.

$$\mathrm{Scene\ Striping\ Correction\ Matrix} = \frac{\mathrm{CTDiff}(m,n)\cdot\mathrm{HomFilt}(m,n)}{2}$$

$$\mathrm{Scene\ Striping\ Metric} = \mathrm{abs}\left(\mathrm{CTDiff}(m,n)\cdot\mathrm{HomFilt}(m,n)\right)$$

6. The detector striping metric is the mean of the columns of the scene striping metric. Each of the individual SCA scene striping metric arrays are concatenated to produce a single band array for the detector striping metric. This tells us how stripy a single detector is. The detector striping metric has individual values for each detector except for the first and last detectors. Figure 1 shows an example detector striping metric.

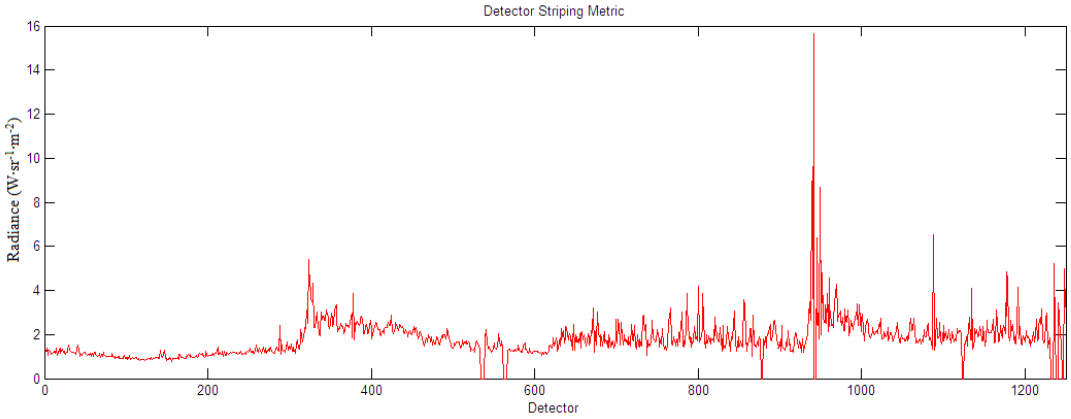


Figure 1: Example Detector Striping Metric.

7. The overall striping metric is derived from the detector striping metric.
   a. First the mean of the entire detector striping metric is found.

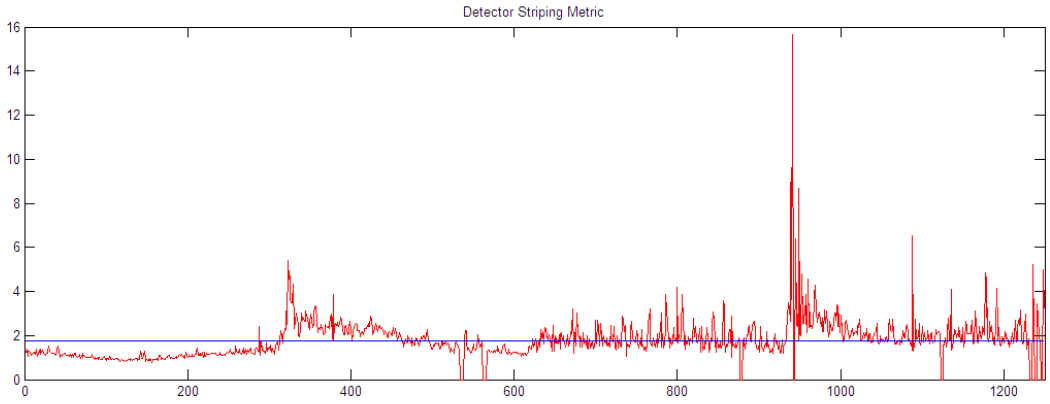


Figure 2: Mean of Detector Striping Metric.

b. Then a 75 length median filter is applied to the detector metric, and smoothed with a 15 length averaging filter.

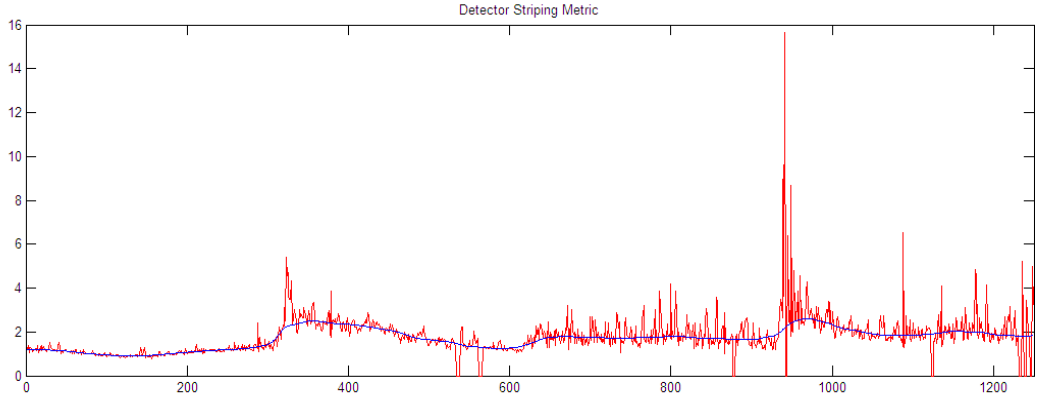


Figure 3: Averaged Median Fit to Detector Striping Metric.

For border detectors on the left side, the median filter will find the median of 37 detectors to the right and however many detectors there are to the left. So for the first detector, it will find the median of the first detector and the 37 detectors to the right. For the second detector it will find the median of the first and second detector and the 37 detectors to the right, and so on until the 38 detector when it find the median of the current detector and 37 detectors to the left and right.
Border detectors on the right side are handled the same way except reversed. So for the last detector it will find the median of the last detector and 37 detectors to the left. The average filter works in a similar way. For the first detector it will take the average of the first detector and 7 detectors to the right.

c. This averaged median fit is subtracted out from the detector striping metric.

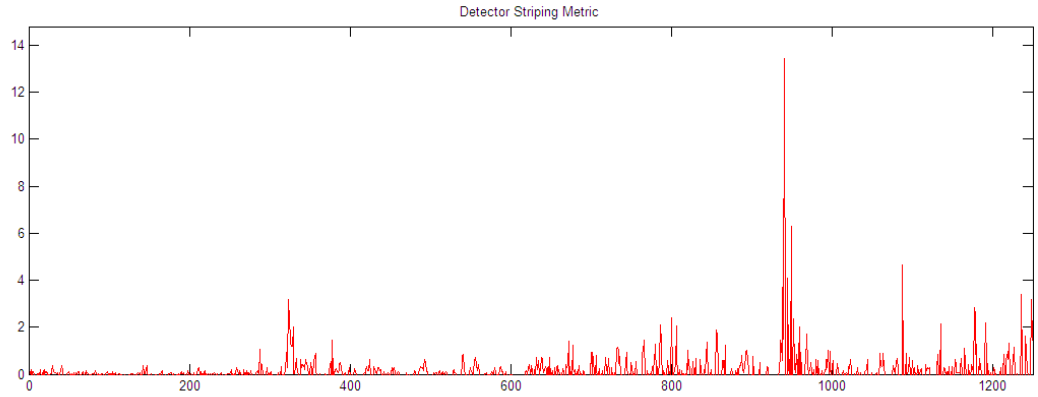


Figure 4: Median Subtracted Detector Striping Metric.

d. The next factor used for the overall striping metric is the maximum peak from this median filter subtracted detector metric.

Figure 5: Maximum of the Median Subtracted Detector Striping Metric.

e. The last factor used is then the mean of the top 15 peaks, including the maximum peak, from the median filter subtracted detector metric. (There are only six peaks circled, but the algorithm should find 15).

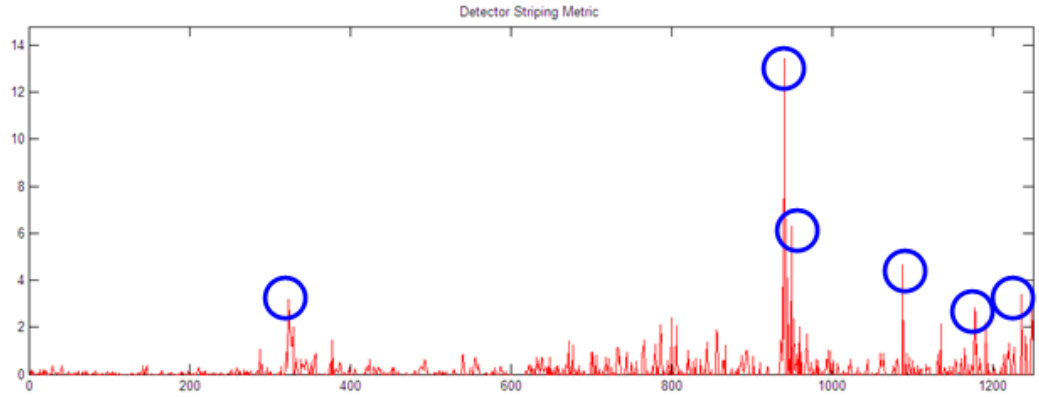


Figure 6: Top Peaks of the Median Subtracted Detector Striping Metric.

It is important to find the top 15 individual peaks. Detectors part of a higher spike should not be used. A detector's two adjacent detectors are considered for determining peaks. If detector *x* has a neighboring detector with a higher value, detector *x* is not a peak.

There is no amount a peak must be larger then its neighbors; it must only be larger. One approach to do this is to arrange the detector striping metric numbers in descending order while maintaining the detector to which the metric numbers correspond. Then one can go down the list and if there is neighboring detector above the current detector, the current detector is not an individual peak. Figure 7 shows this more clearly.

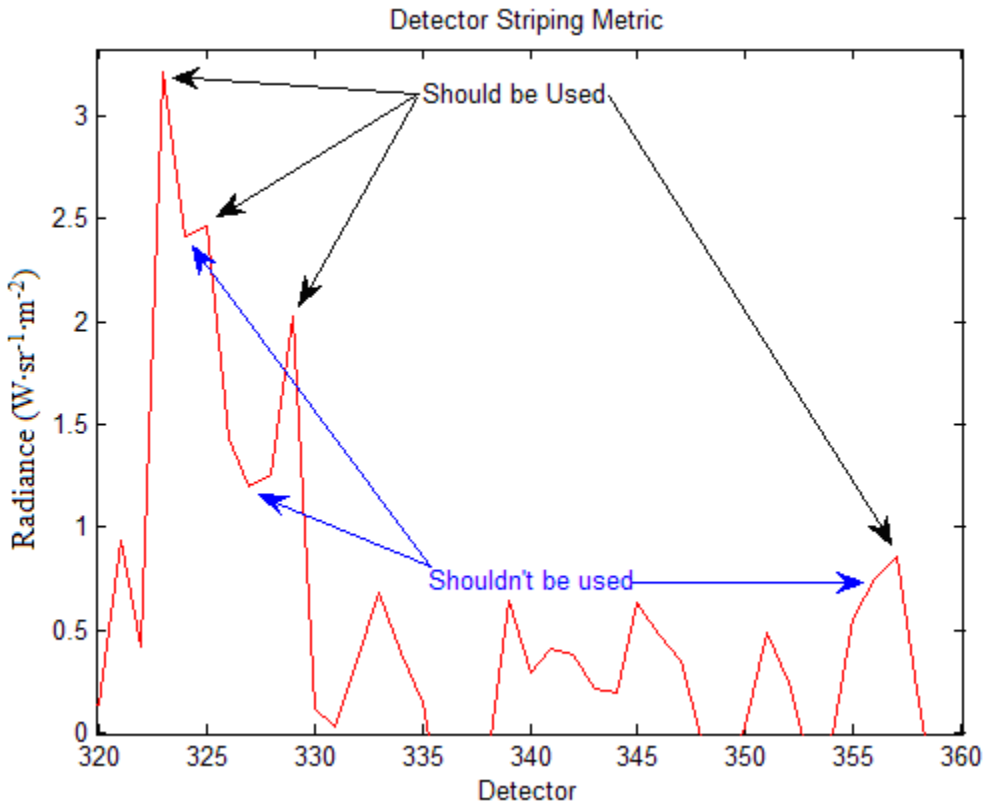


Figure 7: Individual Peak Detectors.

f. The overall striping metric is the cube root of the product of the mean, maximum peak, and mean of the top 15 peaks. This number will be in radiance units ($W \cdot sr^{-1} \cdot m^{-2} \cdot \mu m$). It is also desired to capture this value in DN, so it will have to be backed out of radiance space.

$$\text{Overall Striping Metric} = \sqrt[3]{\text{mean} \cdot \text{max} \cdot \text{mean of top fifteen}}$$

- The mean of the detector striping metric measures the amount of striping present throughout the entire image, odd/even striping being the largest portion.
- The worst single detector stripe is measured by the maximum peak.
- The mean of the top 15 peaks measures the amount of single detector striping throughout the image.

8. If the write striping metric option is on, write the overall striping metric to the database.
9. If the summary report option is on write the overall metric, scene striping metric and detector striping metric to a report.
10. Repeat all steps for all SCAs and Bands.

7.4.11 Non-uniformity Characterization

7.4.11.1 Background

Streaking, Banding, and Full Field Of View (FFOV) Uniformity characterizations provide 4 different measures of detector uniformity. All characterizations will generate metrics for uniformity and stability of uniformity assessments, both pre-launch and post-launch. In the latter case, these will be characterizations will be used for Key Performance Requirement verification.

FFOV Uniformity: the standard deviation of all detector column average radiances across the FFOV within a band shall not exceed 0.5% of the average radiance

There are 2 methods for characterizing Banding.
Method A: The root mean square of the deviation from the average radiance across the full FOV for any 100 contiguous detector column averages of radiometrically corrected OLI image data within a band. This banding specification is met when the metric is less than or equal to 1.0% for OLI and 0.5% for TIRS of the band average radiance.

Method B: The standard deviation of the radiometrically corrected values across any 100 contiguous detector column averages of OLI image data within a band. This banding requirement is met when the metric is less than or equal to 0.25% for OLI and 0.5% for TIRS of the average radiance across the 100 detector columns.

Streaking is measured across any 3 contiguous detector column averages, across the FOV. The streaking requirement is met when the metric is less than 0.005 for OLI bands 1-7 and 9, and 0.01 for the OLI panchromatic band or 0.005 for the TIRS bands. The streaking parameter is defined below.

For OLI, this algorithm is intended to work primarily on solar scenes, though the capability to process uniform earth scenes should be included; verification of the requirements at 2*Ltyp will require extra analysis. For TIRS, this algorithm should be run on blackbody scenes with a temperature set point between 260 and 330K .

Based on the current process flow: These characterizations should be performed on radiance data. The Histogram Statistics Characterization module is performed on bias-corrected and linearized image data, but before the gains and relative gains have been applied. The non-uniformity characterization will not operate on the image data, but rather will use the Histogram Statistics in the database. Therefore, the algorithm will need to apply the gains from the database/CPF to convert the histogram means to radiance.

Analysis of the output data will determine whether, initially, the Non-Uniformity specification is being met and then, later, whether there are changes in uniformity across the focal plane. This algorithm requires a uniform scene or a scene of known non-uniformity. For OLI, the only target that is expected to meet this requirement is the solar diffuser. Therefore this algorithm may only be useful for characterizing the non-uniformity performance, particularly the full field of view uniformity, on one spectral target, as opposed to the three indicated in the requirement. For TIRS, the blackbody will be operated at multiple temperature set points. Each of these blackbody images should be useful in characterizing the banding and streaking over the range of typical Earth temperatures

Note: the notation in this version of the banding equations has been modified from the OLI Requirements Document for clarification.

7.4.11.2 Inputs

The inputs to this algorithm come from either the output of other algorithms (DB) or from a set of calibration parameters (CPF). Table 12 lists the inputs of this algorithm.

Table 12: Algorithm Inputs

Descriptions	Symbol	Units	Level	Source	Type
Detector means (bias corrected, linearized only)	Q_i	DN	N_{band} x N_{SCA} x N_{det}	DB (histogram statistics table)	float
Gains	G	DN/(w/m^2 sr um)	N_{band}		float
Relative gains	r_i	[]	N_{band} x N_{SCA} x N_{det}	CPF	float
Inoperable detectors, out-of-spec detectors			N_{band} x N_{SCA} x N_{det}	CPF	integer
Solar or blackbody non-uniformity scaling factors	ν_i	[]	N_{band} x N_{SCA} x N_{det}	CPF	float

7.4.11.3 Outputs

The outputs of this algorithm are typically stored in the characterization database. However, an option to store this data to an ASCII text file is needed to support testing. This reduces inserts into the database as well as speeding up calibration updates. Table 13 lists the outputs of this algorithm.

Table 13: Algorithm Outputs

Descriptions	Symbol	Units	Level	Target	Type
BandingMetric_FFOV	B_{FFOV_i}_percent	%	N_{band} x N_{SCA} x N_{det}	Db, report	float
BandingMetric_per100pix	$B_{per100det_i}$_percent	%	N_{band} x N_{SCA} x N_{det}	Db, report	float

Full FOV Uniformity Metric	FFOV_metric_percent	%	N_{band}	Db, report	float
Streaking Metric	S_i	[]	N_{band} x N_{SCA} x N_{det}	Db, report	float

7.4.11.4 Options

- Apply non-uniformity scaling factors
- Output ASCII text file summary in addition to reporting data to the database

7.4.11.5 Procedure

For each solar collect, for each band:

1. Determine per-detector radiances (L_i') as given in Equation 1 for selected scene by applying the per-detector gains to the per-detector scene means (Q_i). .

   a. $$L_i' = \frac{Q_i}{G \times r_i} \quad (1)$$

      i. where *G* is the band-average gain, r_i is the per-detector relative gain and *i* is the detector counter. In this algorithm, it is meant to count across the entire focal plane, not just across a single SCA.

   b. An option would be to calculate radiance straight out of the database operation. For example *select histogram_mean / detector_gain from hist_stats, cpf where cpf.date=now() and hist_stats.detector = cpf.detector*

2. For solar and blackbody data: Correct per-detector radiance for non-uniformity (Equation 2) of the solar panel or the blackbody using the per-detector scaling factor (ν_i). For non-solar, non-blackbody data, the scaling factors are set to 1.0 for all detectors.

   a. $$L_i = \frac{L_i'}{n_i} \quad (2)$$

   b. Note: the solar uniformity values in the CPF may be in terms of per-detector reflectance. If that is the case, the reflectances should be normalized to the average before applying them to the radiance.

   c. Note: it is unclear what form the non-uniformity scaling factors will take in the CPF. Once these are defined, we will be able to specify whether the radiances are multiplied or divided by the scaling factors.

3. Stitch the radiance data together in order across the focal plane. Include all imaging detectors. Include overlap detectors. Do not include dark or redundant detectors.

   a. Each SCA will have several detectors that image the same portion of the ground as the adjacent SCA. For example, let's say the SCAs each have 500 detectors and the last 10 detectors of SCA1 image the same ground as the first 10 detectors of SCA2. The radiance array should include both SCA1 detectors 491-500 and SCA2 detectors 1-10.

   b. The redundancy of the overlap detectors should not affect the banding and streaking results of the solar data though it means that we are not measuring the image SCA-to-SCA discontinuity.

4. Calculate Full FOV Uniformity Metric as given below in Equation 3 for the band. Do not include detectors flagged as inoperable or out-of-spec in the calculation.

a. FFOV_metric_percent = stdev(L) / mean(L)*100 (3)

b. Record FFOV_metric_percent to the database or optionally to a file.

5. Calculate banding metrics as given below in Equations 4 and 6 for each imaging detector (i). Do not include detectors flagged as inoperable or out-of-spec in the calculation for operable detectors. Skip the banding calculation for inoperable and out-of-spec detectors.

a. Method 1)

$$B_{FFOV\_i} = \sqrt{\sum_{n=i}^{i+99} (L_n - \bar{L})^2 / 100} \quad (4)$$

$$B_{FFOV\_i\_percent} = \frac{B_{FFOV\_i}}{\bar{L}} * 100 \quad (5)$$

Where:

L_i is the radiance of detector i

$\bar{L}$ is the scene average radiance: $\bar{L} = mean(L)$

b. Method 2)

$$B_{per100\,det\_i} = \sqrt{\sum_{n=i}^{i+99} (L_n - \bar{L}_{100\,det})^2 / 99} \quad (6)$$

$$B_{per100\,det\_i\_percent} = \frac{B_{per100\,det\_i}}{\bar{L}} * 100 \quad (7)$$

Where:

L_i is the radiance of detector i

$\bar{L}_{100\,det}$ is the average radiance across 100 detectors

$$\bar{L}_{100\,det} = \sum_{i=n}^{n+99} L_i / 100 \quad (8)$$

c. Record per-detector banding arrays to database or optionally to a file.

d. In both of these calculations, the calculation cannot be performed for the detectors at the final edge (i.e., the last 99 detectors). As a result of the banding metrics not being associated with the center detector of the window, it is really the first 50 and last 50 detectors that are not characterized. However, it is the banding entries for the last 99 detectors that are left blank.

e. In the case where i…i+99 includes inoperable and/or out-of-spec detectors, the summation should be taken for fewer detectors rather then increasing maintaining the 100 detector average.

6. Calculate streaking metric (Equation 9) for each imaging detector. Do not calculate the streaking metric for detectors flagged as inoperable and out-of-spec. Also, do not calculate streaking metric for detectors adjacent to inoperable or out-of-spec detectors.

$$S_i = \left| L_i - \frac{1}{2}\left(L_{i-1} + L_{i+1}\right) \right| / L_i \quad (9)$$

Where:

L_i is the radiance of detector i;

L_{i-1} and L_{i+1} are similarly defined for the (i-1)th and (i+1)th detector columns.

a. Record per-detector streaking array to database or optionally to a file.

7.4.12 Signal-to-Noise Characterization Noise Equivalent Delta-Temperature Characterization

7.4.12.1 Background

Signal-to-Noise (SNR) characterization and the Noise Equivalent Delta-Temperature (NEΔT) provide an estimate of the overall noise behavior of the OLI and TIRS, respectively. These noise characterizations will be performed both pre- and post-launch. After launch, the SNR characterization will be used for OLI Key Performance Requirement (KPR) verifications and the NEΔT characterization will be used to assess TIRS performance.

TIRS Metrics: For uniform scene temperatures between 240 K and 360 K extending over the full FOV of TIRS, and for a data collection period corresponding to a WRS-2 scene (~ 25 seconds at the nominal frame rate), the median detector standard deviation when converted into radiance units shall be ≤ 0.059 W/m_2 sr µm for the 10.8 µm channel and ≤ 0.049 W/m_2 sr µm for the 12.0 µm channel. This includes quantization noise.

The specification temperatures are defined in Appendix A.

The SNR and NEΔT will be calculated on targets of per-detector uniform radiance, in order to have a good estimate of the noise level at a specific radiance. For OLI the SNR characterization will be performed on the diffuser panel and stim lamp collections, including their associated dark collects. For TIRS, the characterization will be performed on the blackbody and deep space collects.

Some OLI lamp collects will barrel-shifted in order to monitor the lower 12-bits of the instrument. This results in saturation in the SWIR bands (SWIR1, SWIR2, and Cirrus). The results from these bands should be removed from the analysis.

These characterizations are performed on bias-corrected and linearized Histogram Statistics Characterization algorithm generated means and standard deviations (in units of digital number) stored in the IAS database; they do not require separate analyses of image data. However, the SNR specification is written in terms of radiance, so for the final SNR report, either the SNR model or the spec values need to be converted to radiance using the appropriate gains and/or relative gains. Similarly for TIRS data, the noise model must be converted to radiance and temperature. The temperature conversion will require use of a Planck equation approximation; a module exists from Landsat thermal band processing (external to the IAS) but requires the relative spectral response function of the TIRS bands as an input.

Analysis of the output data will determine whether the SNR requirement is being met and as time progresses, whether there are changes in SNR across the focal plane.

7.4.12.2 Dependencies

Histogram Statistics Characterization
(Lamp Characterization)

(Diffuser Characterization)

Inputs

The inputs to this algorithm come from either the output of other algorithms (DB) or from a set of calibration parameters (CPF). Table 12 lists the inputs of this algorithm.

Table 14: Algorithm Inputs

Descriptions	Symbol	Units	Level	Source	Type
Scene identifiers, date, time, lamp pair, diffuser			1	DB	
Detector means for illuminated collects (bias corrected, linearized only)	Q_i	DN	N_{band} x N_{SCA} x N_{det}	DB (histogram statistics table)	float
Detector standard deviation for illuminated collects	σ_i	DN	N_{band} x N_{SCA} x N_{det}	DB (histogram statistics table)	float
Detector standard deviation for paired dark collect	$\sigma_{0,i}$	DN	N_{band} x N_{SCA} x N_{det}	DB (histogram statistics table)	float
Number of valid frames for illuminated collect	N_{valid}	count	N_{bands} x N_{SCAs} x $N_{detectors}$	Db (histogram statistics table)	Int
Number of valid frames for paired dark collect	N_{valid_dark}	count	N_{bands} x N_{SCAs} x $N_{detectors}$	Db (histogram statistics table)	Int
Gains	G	$DN/(W/m^2$ sr um)	N_{band}		float
Relative gains	r_i	[]	N_{band} x N_{SCA} x N_{det}	CPF	float
Inoperable detectors, out-of-spec detectors			N_{band} x N_{SCA} x N_{det}	CPF	integer
Relative Spectral Response curves for TIRS bands	RSR	[]			

7.4.12.3 Outputs

The outputs of this algorithm are typically stored in the characterization database. However, an option to store this data to an ASCII text file is needed to support testing. This reduces inserts into the database as well as speeding up calibration updates. Table 13 lists the outputs of this algorithm.

Table 15: Algorithm Outputs

	Symbol	Units	Level	Target	Type
Scene identifiers, date, time, lamp pair, diffuser			1	DB	
Mean Signal Level of illuminated collect (duplicate of histogram stat input)	$\overline{Q}$	DN	N_{band} x N_{SCA} x N_{det}	IDL save file, report	float
Signal standard deviation of illuminated collect (duplicate of histogram stat input)	σ_i	DN	N_{band} x N_{SCA} x N_{det}	IDL save file, report	float
Signal level of paired dark collect (assume 0 for OLI, but use real data for TIRS)	Q_0	DN	N_{band} x N_{SCA} x N_{det}	IDL save file, report	float
Standard deviation of paired dark collect	$\sigma_{0,i}$	DN	N_{band} x N_{SCA} x N_{det}	IDL save file, report	float
Number of valid frames (duplicate of histogram stat input)	N_{valid}	count	N_{band} x N_{SCA} x N_{det}	DB, IDL save file, report	int
Number of valid frames for paired dark collect	$N_{valid\ dark}$	count	N_{bands} x N_{SCAs} x $N_{detectors}$	DB, IDL save file, report	Int
Noise model coefficients	a, b	[DN], [DN^2]	N_{bands} x N_{SCAs} x $N_{detectors}$	DB, IDL save file, report	float
Uncertainty in Noise model coefficients	unc_a, unc_b	[DN], [DN^2]	N_{bands} x N_{SCAs} x $N_{detectors}$	DB, IDL save file, report	float
Relative Uncertainty in Noise model coefficients	rel_unc_a, rel_unc_b	[DN], [DN^2]	N_{bands} x N_{SCAs} x $N_{detectors}$	IDL save file, report	float
OLI: SNR at spec levels (Ltyp and Lhigh)	SNR_{Ltyp}, SNR_{Lhigh}	[]	N_{bands} x N_{SCAs} x $N_{detectors}$	DB, IDL save file, report	float
OLI: uncertainty in SNR at spec levels	unc_{SNR}	[]	N_{bands} x N_{SCAs} x $N_{detectors}$	DB, IDL save file, report	float
TIRS: NEDT at spec	NEDT	K	N_{bands} x	IDL save	float

levels (5 temperatures)			N_{SCAs} x $N_{detectors}$ N_{bands} x	file, report	
TIRS: uncertainty at spec levels	unc_{NEDT}	K	N_{SCAs} x $N_{detectors}$ N_{bands} x	IDL save file, report	float
TIRS: NEDL at a single spec temperature	NEDL	[radiance]	N_{SCAs} x $N_{detectors}$ N_{bands} x	DB	float
TIRS: uncertainty in NEDL at spec temperature	unc_{NEDL}	[radiance]	N_{SCAs} x $N_{detectors}$	DB	float
Band-average Mean signal	Q_{BA}	DN	N_{bands}	DB	float

7.4.12.4 Options

- Report data to ascii report files as well as IDL save file and plots.

7.4.12.5 Procedure

For each appropriate uniform collect, for each band:

7. Extract Histogram Statistics database table for each detector; Q_i, σ_i, $Q_{0,i}$, $\sigma_{0,i}$, N_{valid}, and N_{valid_dark}. Extract gains and relative gains from the CPF database table: G and r.
   a. Queries will generate a list of solar, lamp and blackbody collects acquired since the last time the query was run. A second query will generate the database output necessary to run snr.pro for all new collects.
   b. For the SWIR1, SWIR2 and Cirrus bands, filter the lamp collections with the lower 12-bit barrel shift. These bands have an additional filter in the query for the truncation flag: only extract data from the database if the truncation flag is 0. The 12-bit barrel shifted data are included in the data set for the other bands.
8. Calculate band-average means from database query output:
   i. Q_{BA} = mean(Q_i)
   ii. $Q_{0,BA}$ = mean($Q_{0,i}$)
9. Calculate Signal-to-Noise ratio for each detector
   a. $SNR_i = \frac{Q_i}{s_i}$ (1)
10. Calculate noise model coefficients using illuminated data and paired dark data. The equation for the noise model is $s_i^2 = a + b * Q_i$
    a. $a = \sigma_{0,i}^2$ (2)
    b. $b = \frac{\sigma_i^2 - a}{Q_i}$ (3)
11. Calculate uncertainty on the fit coefficients

a. $unc_a = \frac{2a}{\sqrt{2*(N_{valid\_dark}-1)}}$ (4)

b. $unc_{Q_i} = \frac{\sigma_i}{\sqrt{N_{valid}}}$ (5)

c. $unc_{\sigma_i} = \frac{2\sigma_i}{\sqrt{2*(N_{valid}-1)}}$ (6)

d. $unc_b = b * \sqrt{\left(\frac{unc_{Q_i}}{Q_i}\right)^2 + \left(\frac{\sqrt{unc_a^2+unc_{\sigma_i}^2}}{{\sigma_i}^2-a}\right)^2}$ (7)

12. FOR OLI BANDS:

a. Convert Ltyp and Lhigh to DN using the CPF for the scene.

i. $Q_{typ} = G * r * L_{typ}$ (8)

ii. $Q_{high} = G * r * L_{high}$ (9)

b. Calculate SNR at Ltyp and Lhigh for each detector using the fit coefficients

i. $SNR_{L_{typ}} = \frac{DN_{typ}}{\sqrt{a+b*DN_{typ}}}$ (10)

ii. $SNR_{L_{high}} = \frac{DN_{high}}{\sqrt{a+b*DN_{high}}}$ (11)

c. Calculate the uncertainties on the spec SNR values (propagation of error).

i. $\frac{\partial SNR}{\partial a} = \frac{-1*SNR}{2*(a+b*DN)}$

ii. $\frac{\partial SNR}{\partial b} = \frac{-1*DN*SNR}{2*(a+b*DN)}$

iii. $unc_{SNR} = \sqrt{\left(\frac{\partial SNR}{\partial a} unc_a\right)^2 + \left(\frac{\partial SNR}{\partial b} unc_b\right)^2 + 2\rho \frac{\partial SNR}{\partial a}\frac{\partial SNR}{\partial b} unc_a unc_b}$

iv. where ρ is the correlation coefficient on the regression of Q and noise[2], which will always be 1, since there are only two points in the regression

d. There are cases in the pre-launch data where the dark noise is greater than the noise in the illuminated data. These data result in an SNR model that is negative (b is negative), and the SNR at Lhigh was output as –NaN, which the database would not ingest. In the cases where b is less than zero, all SNR parameters are set to -999, a bad-value flag.

e. Count the number of detectors that do not meet specification and those that are less than 80% of specification, taking into account the uncertainty for each detector.

i. $out\_of\_spec\_index = where(SNR + 2 * unc_{SNR} < spec\_value)$

ii. Note that while this algorithm will detect out-of-spec detectors, the responsibility of counting and reporting them falls to the Detector Operability algorithm, via the data in the SNR database table.

f. Assess if the median SNR at the Ltyp meets specification.

13. FOR TIRS bands:

a. Calculate the slope and offset (m_{planck} and b_{planck}) of the Planck equation for the specification temperatures using planck.pro, a module written long ago for Landsat thermal band processing. [requires a spectral curve for each band]

b. Convert specified temperature (see Appendix C) to counts:

i. $DN_T = \left(T_{spec} * m_{planck} + b_{planck}\right) * G$

c. Calculate NEDL and NEDT and uncertainties at each specified temperature

i. $NEDL = \frac{\sqrt{a+b*DN_T}}{G}$

ii. $NEDT = \frac{NEDL}{m_{planck}}$

iii. $\frac{\partial noise}{\partial a} = \frac{1}{2*\sqrt{a+b*DN_T}}$

iv. $\frac{\partial noise}{\partial b} = \frac{DN_T}{2*\sqrt{a+b*DN_T}}$

v. $unc_{noise} = \sqrt{\left(\frac{\partial noise}{\partial a} unc_a\right)^2 + \left(\frac{\partial noise}{\partial b} unc_b\right)^2 + 2\rho \frac{\partial noise}{\partial a} \frac{\partial noise}{\partial b} unc_a unc_b}$

d. Assess the NEDT at each temperature against the specification requirements. Count and flag failing detectors, taking into account the uncertainty for each detector.

i. $out\_of\_spec\_index = where(NEDT - 2 * unc_{noise} > spec\_value)$

14. Output SNR, fit coefficients and uncertainties for every detector to IDL save file and make associated plots. See below for plots and Appendix B for output reports.

15. Ingest the SNR and NEDL ascii files into the SNR Characterization and NEDL Characterization tables.

7.4.12.6 Maturity

The code is meant for the IAS tool box and thus, is not robust and hands-off as are the usual IAS algorithms. I expect that I will watch the output pop-up as I run this on a monthly basis and will add and subtract functionality as I see fit.

Appendix A. SNR Requirements

Table 1. Radiance Levels for SNR Requirements (from OLI Requirements Document)

#	Band	Radiance Level for SNR, L (W/m^2 sr µm)		Saturation Radiances, L_{Max} (W/m^2 sr µm)
		Typical, $L_{Typical}$	High, L_{high}	Requirement
1	Coastal Aerosol	40	190	555
2	Blue	40	190	581
3	Green	30	194	544
4	Red	22	150	462
5	NIR	14	150	281
6	SWIR 1	4.0	32	71.3
7	SWIR 2	1.7	11	24.3
8	Panchromatic	23	156	515
9	Cirrus	6.0	N/A	88.5

Table 2. SNR Requirements

#	Band	SNR Requirements	
		At $L_{Typical}$*	At L_{High}*
1	Coastal Aerosol	130	290
2	Blue	130	360
3	Green	100	390
4	Red	90	340
5	NIR	90	460
6	SWIR 1	100	540
7	SWIR 2	100	510
8	Panchromatic	80	230
9	Cirrus	50	N/A

Table 3. NEDT Requirements

	NEDT at 240K	NEDT at 260K	NEDT @ 300K	NEDT @ 320K	NEDT @ 360K
TIRS1	0.80	0.61	0.40	0.35	0.27
TIRS2	0.71	0.57	0.40	0.35	0.29

Appendix B

Sample output text files. There is a summary file reporting if the band median numbers meet specification and a per-band file reporting the individual detectors that do not meet specification.

```
ibarsi@biatov% more SNR.SpecSummary.dat
      Band     Median SNR@Ltyp    Requirement    Median SNR@Lhigh    Requirement  MedianLtypOOS  NDetsOOS@Ltyp NDetsOOS@Lhigh
        CA             256.146            130             628.632            290                              0              0
      Blue             431.395            130            1223.478            360                              0              0
     Green             366.482            100            1322.959            390                              0              0
       Red             274.935             90            1037.538            340                              0              0
       NIR             238.194             90            1069.464            460                              0              0
     SWIR1             296.007            100            1045.821            540                              0              0
     SWIR2             357.874            100            1049.833            510                              0              0
    Cirrus             172.688             50            2725.180             50                              0              0
       Pan             158.277             80             454.908            230                              0              0


ibarsi@biatov% more CA SNR.SpecWarnings.dat
  Detector  SNR@Ltyp    Uncertainty    Requirement SNR@Lhigh    Uncertainty    Requirement    Out-of-Spec
      6209    5.9068           0.05            130   12.8743           0.05            290              Y


ibarsi@biatov% more SNR.SpecSummary.dat
      Band     T    Median NEDL@L(T)    Requirement    Median NEDT@L(T)    Requirement  MedianL(T)OOS  NDetsOOS@L(T)
     TIRS1  240               0.012          0.059               0.159          0.809                              1
     TIRS1  260               0.012          0.059               0.126          0.619                              1
     TIRS1  300               0.013          0.059               0.090          0.414                              1
     TIRS1  320               0.013          0.059               0.080          0.355                              1
     TIRS1  360               0.014          0.059               0.068          0.278                              1

      Band     T    Median NEDL@L(T)    Requirement    Median NEDT@L(T)    Requirement  MedianL(T)OOS  NDetsOOS@L(T)
     TIRS2  240               0.015          0.049               0.215          0.720                              0
     TIRS2  260               0.015          0.049               0.173          0.571                              0
     TIRS2  300               0.015          0.049               0.127          0.404                              0
     TIRS2  320               0.016          0.049               0.113          0.354                              0
     TIRS2  360               0.016          0.049               0.094          0.288                              0


ibarsi@biatov% more TIRS1 SNR.SpecWarnings.dat
  Detector      NEDL       Req   SetTemp     NEDT+/-unc          ReqOut-of-Spe
      1724    0.1269    0.0590       240    1.7410+/-0.5315    0.8094         Y
      1724    0.1568    0.0590       260    1.6464+/-0.4067    0.6194         Y
      1724    0.2205    0.0590       300    1.5464+/-0.2717    0.4138         Y
      1724    0.2535    0.0590       320    1.5237+/-0.2329    0.3547         Y
      1724    0.3203    0.0590       360    1.5108+/-0.1828    0.2783         Y
```

7.4.13 Detector Operability Characterization

7.4.13.1 Background

Per the LDCM Acronym List and Lexicon (GSFC 427-02-06):

A detector is considered operable, even if out of spec, when it meets the following requirements:

a. The detector is sensitive to photons within its spectral band and not saturated at expected operating temperatures under dark conditions.

b. The detector's noise is less than 5 times the mean noise level for the band on which it occurs.

c. The detector's dark current remains within +/- 5 times the RMS noise over the period between dark frame references.

d. The detector's actual dynamic response range is greater than 25% of the specified dynamic range; such that the Actual Dynamic Range >= 0.25 * Specified Dynamic Range.

For OLI, requirement “a.”, the phrase “detector is sensitive to photons” was refined to “detector’s count/radiance slope is greater than 20% of the mean count/radiance slope for the band”

This algorithm determines the operability status of OLI and TIRS detectors based on these criteria, and in addition, for two of these criteria, noise level and dynamic range, assesses whether operable detectors are also within specifications. An additional complication is that “dynamic range” is not used in the OLI requirements document. Saturation radiance, though not strictly the same, is used in place of dynamic range.

Determination of the operability status of the detectors requires data from histogram statistics that are stored in the characterization database. The statistics derived from specific scene types (e.g. shutter and solar) will be used to derive various metrics including total noise and dynamic range. The signal-to-noise ratio (SNR) and noise equivalent delta radiance (NEdL) are other derived statistics that are useful for detector operability characterization. The SNR values for OLI and NEdL for TIRS for each detector will come from the SNR/NEdL characterization algorithm, which itself requires data from histogram statistics.

Based upon the trending and evaluation of these metrics, a per-detector status will be assigned describing by which criteria the detector was deemed inoperable.

Finally, the detector operability status will be used to populate and validate the per detector operability flags in the calibration parameter file. The process to do any updates is currently manual after review by an analyst. When a detector is flagged as inoperable, the data for that detector is replaced prior to producing the final radiometric product (see Correct Dead Detectors ADD).

7.4.13.2 Dependencies

Histogram Statistics Characterization
SNR Characterization

Input

Descriptions	Symbol	Units	Level	Source	Type
Minimum	Q_{min}	DN	N_{bands} x N_{SCAs} x $N_{detectors}$	Histogram Statistics	Float
Maximum	Q_{max}	DN	N_{bands} x N_{SCAs} x $N_{detectors}$	Histogram Statistics	Float
Mean	$\overline{Q}$	DN	N_{bands} x N_{SCAs} x $N_{detectors}$	Histogram Statistics	Float
Standard deviation	σ	DN	N_{bands} x N_{SCAs} x $N_{detectors}$	Histogram Statistics	Float
Signal-to-noise ratio	*SNR*	*n/a*	N_{bands} x N_{SCAs} x $N_{detectors}$	SNR Char.	Float
Band mean SNR	$\overline{SNR}$	*n/a*	N_{SCAs} x $N_{detectors}$	SNR Char.	Float
Within-band standard deviation of SNR	σ_{SNR}	*n/a*	N_{SCAs} x $N_{detectors}$	SNR Char.	Float
Signal-to-noise ratio uncertainty	SNR_{unc}	*n/a*	N_{SCAs} x $N_{detectors}$	SNR Char.	Float
Relative Gain	G_{rel}	*n/a*	N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Float

Absolute Gains	G_{abs}	$\frac{DN}{W/m^2 sr\mu m}$	N_{band} x N_{SCA}	CPF	Float
Noise equivalent delta radiance	*NEdL*			NEDL Char.	Float
Noise equivalent delta radiance uncertainty	$NEdL_{unc}$				

7.4.13.3 Output

Description	Level	Target	Type
Detector Status	N_{bands} x N_{SCAs} x $N_{detectors}$	Db	Int

7.4.13.4 Procedure

The procedure for marking detectors as inoperable or out-of-spec is a simple gamut of threshold tests. In order to keep track of all of the ways that a detector could be marked inoperable, the detectors will be assigned a status. The status is a 16-bit number, where each bit represents a different reason for which a detector could be flagged (see table).

Table 3. Summary of Operability Metrics

Metric	Equation Number	Specification Status Bit	KPR
Inop- Too Noisy	1	0	12
Inop- Weak Response	2	1	12
Out-of-Spec - Noise	3	2	12, 13
Inop- No Response	4	3	12
Inop-No Response, No Noise	5	4	12
Inop-No Response, Saturated	6	5	12
Inop-Low Saturation Radiance	8	6	12
Out-of-Spec – Saturation Radiance	9	7	13
Inop-unstable dark	10	8	12
Inop- noisy (TIRS only)	11	9	-
Out-of-spec- (TIRS only)	12	10	-

7.4.13.4.1 Noisy Response

As stated in the requirements, a detector is flagged as inoperable if noise greater than 5 times the mean in-band noise level.

$$\sigma(b,s,d) > 5\sigma_{avg}(b) \quad (1)$$

where *b* is band, *s* is sensor chip assembly (SCA) and *d* is detector. If a detector is flagged for being too noisy by this metric, the specification status bit for this metric will be set to 1.

7.4.13.4.2 Unresponsive

As stated in the requirements, a detector should be flagged as inoperable if the count/radiance slope is less than 20% of the mean count/radiance slope for that band.

$$g(b,s,d) < 0.2g_{avg}(b) \quad (2)$$

However, since OLI has a much higher signal to noise ratio than was required. Therefore, it is possible for a detector to fail this requirement, but still be acceptable. Detectors flagged by (2) will be considered out of spec and the specification status bit for this metric will be set to 1. In order to evaluate the responses of the detectors to light and make sure that they are behaving acceptably, a detector will be declared inoperable and the status bit for this metric (unresponsive) will be set to 1 if the detector's SNR is less than the 20% of the band median specified SNR as stated in the requirements. If less than 80% of the band median specified SNR, the detector flag of out-of-spec for SNR will be set to 1. The SNR requirements can be found in Table 1. Note this check is done only for OLI detectors.

$$SNR(b,s,d) + 2SNR_{unc}(b,s,d) < SNR_{req}(b) \quad (3)$$

Table 1. SNR Requirements

Band	SNR Requirements (band median)		Out-of-Spec SNR (80%)		Inoperable SNR (20%)	
	At $L_{Typical}$	At L_{High}	At $L_{Typical}$	At L_{High}	At $L_{Typical}$	At L_{High}
1	130	290	104	232	26	58
2	130	360	104	288	26	72
3	100	390	80	312	20	78
4	90	340	72	272	18	68
5	90	460	72	368	18	92
6	100	540	80	432	20	108
7	100	510	80	408	20	102
8	80	230	64	184	16	46
9	50	N/A	40	N/A	10	N/A

A detector is flagged as inoperable if the detector is no longer sensitive to photons. This is evident when the detector's DN value never changes beyond random noise levels(4), or has no signal whatsoever (5).

$$Q_{\max}(b,d,s) - Q_{\min}(b,s,d) < 3\sigma(b,s,d) \quad (4)$$

$$Q_{\max}(b,s,d) = Q_{\min}(b,s,d), \sigma(b,s,d) = 0 \quad (5)$$

A detector is flagged as Inoperable if the detector is saturated at expected operating temperatures under dark conditions. The saturation value is (1111 1111 1111 binary) in 12-bit space, but this procedure is run on data that has been barrel-shifted into 14-bit space. The nature of the barrel-shifting fills the lower bits with zero, so the saturation value is (11 1111 1111 1100 binary) in 14-bit space, or Q_{sat} = 16,380 DN.

$$Q_{\min}(b,s,d) \geq Q_{sat} \quad (6)$$

7.4.13.4.3 Dynamic Response[Saturation Radiance]

Dynamic response is the range of radiances that a detector is capable of detecting. In addition, a detector is flagged as inoperable if the detector's actual dynamic response is less than 25% of the maximum radiance that it should detect. To determine a detector's operability in this sense, the per-detector gains are applied to Qsat to estimate the radiances at which each detector saturates.

$$L_{sat}(b,s,d) = \frac{Q_{\max}(b,s,d) - \overline{Q}(b,s,d)}{G_{rel}(b,s,d)G_{abs}(b,s)} \quad (7)$$

This value is then compared 25% of the maximum radiance that a detector should detect (L_{max}) and declared inoperable if it is less than that value. If the detector is flagged as inoperable, the status bit for that metric is set to 1

$$L_{sat}(b,s,d) < 0.25 L_{\max}(b) \quad (8)$$

A detector is defined as out of spec if the detector's saturation radiance is less than L_{max}. If the detector is flagged as out of spec, the status bit for that metric is set to 1.

$$L_{sat}(b,s,d) < L_{\max}(b) \quad (9)$$

7.4.13.4.4 Dark Current

Dark Current is equivalent to the mean DN reported by a detector during a dark collection, as measured by Histogram Statistics. The drift of dark current is measured as the absolute difference between the averages of two dark collects. A detector is flagged as inoperable of the drift is greater than 5 times that detector's noise.

$$\left|\overline{Q_1}(b,s,d) - \overline{Q_2}(b,s,d)\right| > 5\sigma(b,s,d) \quad (10)$$

7.4.13.4.5 Thermal Noise Equivalent Delta Radiance (TIRS Only)

For TIRS, one detector operability test is a check of the median per detector standard deviation of uniform on-board calibrator scenes in terms of radiance against the noise equivalent delta radiance (NEdL) threshold listed in the table.

$$median\left(\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(L_i(b,s,d) - \overline{L(b,s,d)}\right)^2}\right) > NEdL_{thresh}(b) \qquad (11)$$

Table 2. NEdL Requirements

TIRS Band	NEdL threshold radiance
1	0.059
2	0.049

Also, a TIRs detector is flagged as out of spec if the NEdL minus twice the noise uncertainty is greater than the 5 times the spec value.

$$NEdL(b,s,d) - 2NEdL_{unc}(b,s,d) > 5NEdL_{thresh}(b) \qquad (12)$$

7.4.14 Relative Gain Characterization (Histogram Method)

7.4.14.1 Background

This function calculates the relative gain of a detector for a given band and SCA from the lifetime scene histogram statistics. There are four different algorithms to calculate the relative gain: classical average mean, classical average standard deviation, SMA-1, and SMA-2.

The relative gains obtained from any method can be applied to the image to correct the striping due to differences in detector response. Some combination of methods may also be used to generate a single set of relative gains.

7.4.14.2 Inputs

Descriptions	Level	Type
Histogram Statistics		
Detector Mean, $\overline{Q}_{\det}$	Nband, Nsca, Ndet	Float
Detector Standard Deviation, σ_{det}	Nband, Nsca, Ndet	Float
Adjacent Detector Correlation, ρ	Nband, Nsca, Ndet	Float
Number of Valid Frames, $\#_{frames}$	Nband, Nsca, Ndet	Int

Threshold values		
Minimum Mean	Nband, Nsca	Int or Float
Maximum Mean	Nband, Nsca	Int or Float
Minimum Standard Deviation	Nband, Nsca	Int or Float
Maximum Standard Deviation	Nband, Nsca	Int or Float
Minimum Number of Frames	Nband	Int
Maximum Number of Frames	Nband	Int

7.4.14.3 Outputs

Descriptions	**Level**	**Type**
Relative Gain		
Ratio of Means		
Classical Average	Nband, Nsca, Ndet	Float
Ratio of Standard Deviation		
Classical Average	Nband, Nsca, Ndet	Float
SMA 1	Nband, Nsca, Ndet	Float
SMA 2	Nband, Nsca, Ndet	Float

7.4.14.4 Procedure

1. Read in the processing parameters.
2. Read in an SCA.
3. Determine if each scene in the data interval is valid.
   a. A scene must have data for all SCAs.
   b. Each SCA mean, $\overline{Q}_{SCA}$, for a scene must be within the minimum and maximum mean thresholds.
   c. Each SCA standard deviation, σ_{SCA}, for a scene must be within the minimum and maximum standard deviation thresholds.
   d. The number of frames in a scene must fit within the number of frames thresholds.

   If a scene does not meet each of these conditions it is invalid and its data will not be used.
4. For every scene in the interval, weight each detector mean, $\overline{Q}_{\det}$, standard deviation, σ_{det}, sum of squares, $SumQ^2$, and adjacent detector correlation, ρ with the number of valid frames, $\#_{frames}$.

$$\overline{Q}_{weight} = \overline{Q}_{\det} \cdot \#_{frames} \qquad \sigma_{weight} = \sigma_{\det} \cdot \#_{frames}$$

$$SumQ^2{}_{weight} = SumQ^2 \cdot \#_{frames} \qquad \rho_{weight} = \rho \cdot \#_{frames}$$

5. Calculate global weighted detector statistics by summing each weighted factor in the interval, and dividing that sum by the total number of frames.

$$\overline{Q}_{global} = \frac{\overline{Q}_{weight,1} + \overline{Q}_{weight,2} + \cdots + \overline{Q}_{weight,m}}{\#_{frames,1} + \#_{frames,2} + \cdots + \#_{frames,m}}$$

$$\sigma_{global} = \frac{\sigma_{weight,1} + \sigma_{weight,2} + \cdots + \sigma_{weight,m}}{\#_{frames,1} + \#_{frames,2} + \cdots + \#_{frames,m}}$$

$$SumQ^2{}_{global} = \frac{SumQ^2{}_{weight,1} + SumQ^2{}_{weight,2} + \cdots + SumQ^2{}_{weight,m}}{\#_{frames,1} + \#_{frames,2} + \cdots + \#_{frames,m}}$$

$$\rho_{global} = \frac{\rho_{weight,1} + \rho_{weight,2} + \cdots + \rho_{weight,m}}{\#_{frames,1} + \#_{frames,2} + \cdots + \#_{frames,m}}$$

where 1 and 2 are the first scenes in the interval and *m* is the last.

6. Once the global detector statistics are calculated, the relative gain can be calculated using the different methods.

   a. The SCA average mean method is calculated by taking the ratio between the global detector means and the SCA average mean. The SCA average mean is the mean of all the global detector means in the SCA.

$$\frac{\overline{Q}_{global},1}{\overline{Q}_{global},SCA}, \frac{\overline{Q}_{global},2}{\overline{Q}_{global},SCA}, \cdots, \frac{\overline{Q}_{global},m}{\overline{Q}_{global},SCA}$$

   b. For the SCA average standard deviation method, the gains are calculated by taking the ratio between the global detector standard deviations and the SCA average standard deviation. The SCA average standard deviation is the mean of all the global detector standard deviations in the SCA.

$$\frac{\sigma_{global},1}{\sigma_{global},SCA}, \frac{\sigma_{global},2}{\sigma_{global},SCA}, \cdots, \frac{\sigma_{global},m}{\sigma_{global},SCA}$$

   c. The SMA 1 method gains are calculated by solving a matrix equation that contains the $SumQ^2$ and ρ products.
      i. First the global $SumQ^2$ and ρ products should be put into a matrix as shown below. Besides the last row, the matrix only has nonzero entries along two diagonals.

$$\begin{bmatrix} SumQ^2{}_{global},1 & -\rho_{global},1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & SumQ^2{}_{global},2 & -\rho_{global},2 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & SumQ^2{}_{global},m-1 & -\rho_{global},m-1 \\ 1 & \cdots & \cdots & \cdots & 1 & 1 & 1 \end{bmatrix}$$

where *m* is the number of detectors.

      ii. A reciprocal relative gain matrix is multiplied with this matrix, and their product is set equal to a zero matrix. The last entry in the zero matrix is set to the number of detectors to ensure that the mean of the relative gain estimations will be equal to one.

$$\begin{bmatrix} SumQ^2_{global},1 & -\rho_{global},1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & SumQ^2_{global},m-1 & -\rho_{global},m-1 \\ 1 & \cdots & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{r_1} \\ \vdots \\ \frac{1}{r_{m-1}} \\ \frac{1}{r_m} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m \end{bmatrix}$$

where *r* is the relative gain.

iii. Once these matrices are set up, any matrix solution method can be used to solve the equation.

iv. The element-by-element reciprocal of the relative gains vector will have to be taken to get the relative gains.

d. The SMA 2 method is similar to the SMA 1 method.

i. The global $SumQ^2$ and ρ products should be put into the matrix a little differently as shown below. This (tri-diagonal) matrix has values along three diagonals, except for the first and last rows which only have two entries.

$$\begin{bmatrix} SumQ^2_{global},1 & -\rho_{global},1 & 0 & 0 & 0 & \cdots & 0 \\ -\rho_{global},1 & 2\cdot SumQ^2_{global},2 & -\rho_{global},2 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & -\rho_{global},m-2 & 2\cdot SumQ^2_{global},m-1 & -\rho_{global},m-1 \\ 0 & \cdots & \cdots & \cdots & 0 & -\rho_{global},m-1 & SumQ^2_{global},m \end{bmatrix}$$

ii. A reciprocal relative gain vector is multiplied with this matrix, and their product is set equal to a zero matrix. The zero matrix here is not actually zero but the smallest possible positive non zero number the computer can use.

$$\begin{bmatrix} SumQ^2_{global},1 & -\rho_{global},1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -\rho_{global},m-1 & SumQ^2_{global},m \end{bmatrix} \begin{bmatrix} \frac{1}{r_1} \\ \vdots \\ \frac{1}{r_m} \end{bmatrix} = \begin{bmatrix} \approx 0 \\ \approx 0 \\ \approx 0 \end{bmatrix}$$

iii. Once these matrices are set up, any matrix solution method can be used to solve the equation.

iv. The element-by-element reciprocal of the relative gains vector will have to be taken to get the relative gains.

v. After the relative gains have been found, they will have to be normalized to one.

7. Repeat Steps 2-6 for all SCAs and Bands.

7.4.15 Relative Gain Characterization (90-Degree Yaw)

7.4.15.1 Background

For whiskbroom imaging sensors such as the Landsat TM/ETM+, relative gain estimates for each detector can be derived from the summary statistics of an individual scene. This approach is not feasible for pushbroom imaging sensors such as the LDCM TIRS and the LDCM OLI, because the nominal pushbroom focal plane orientation (orthogonal to the direction of platform motion) does not allow all detectors to measure the same radiance levels in a statistical sense. However, having the platform perform a 90° yaw maneuver re-orients the sensor's focal plane essentially parallel to the direction of platform motion and, with adequate compensation for Earth rotation, allows all detectors to respond to essentially the same radiance levels. Such a maneuver would then permit derivation of relative gain estimates from the summary statistics of an individual scene on an FPM by FPM level.

This algorithm derives relative gain estimates for each operable detector from 90° yaw image data. Given sufficient similarity between the current simulated data and the anticipated flight data, the algorithm can be adapted for use by the LDCM Image Processing Element's Image Assessment Subsystem (IAS).

Figure 1 illustrates a simple flow chart for the estimation of relative gains through use of the 90 degree yaw maneuver. Included in this flow is a step to produce a simulated yaw image, a image manipulation step to properly align detector responses, followed by the user's choice of algorithms for estimation of relative gains—an approach based on the statistics obtained from each detector and a method using the SMA algorithm.

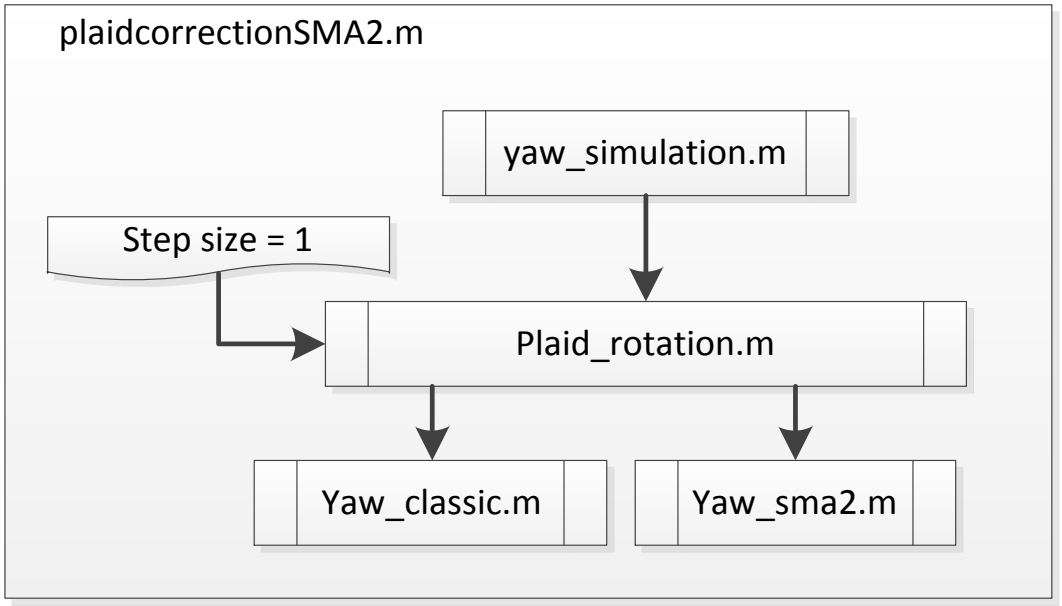


Figure 1. Flow chart for relative gain estimation using the 90 degree yaw maneuver.

7.4.15.2 Input

Descriptions	Level	Source	Type
Yaw Scene	Nband		Float

Output

Descriptions	Level	Destination	Type
Relative Gain	Nband, Nsca, Ndet	Db	Float

7.4.15.3 Procedure

1. Create a simulated yaw image. The linear imaging array for TIRS results in a one pixel lag between adjacent detectors. The offset between even and odd detectors of the OLI focal plane results in a two pixel lag between adjacent even and adjacent odd detectors, which are analyzed separately. Figure 2 shows a simulated image.

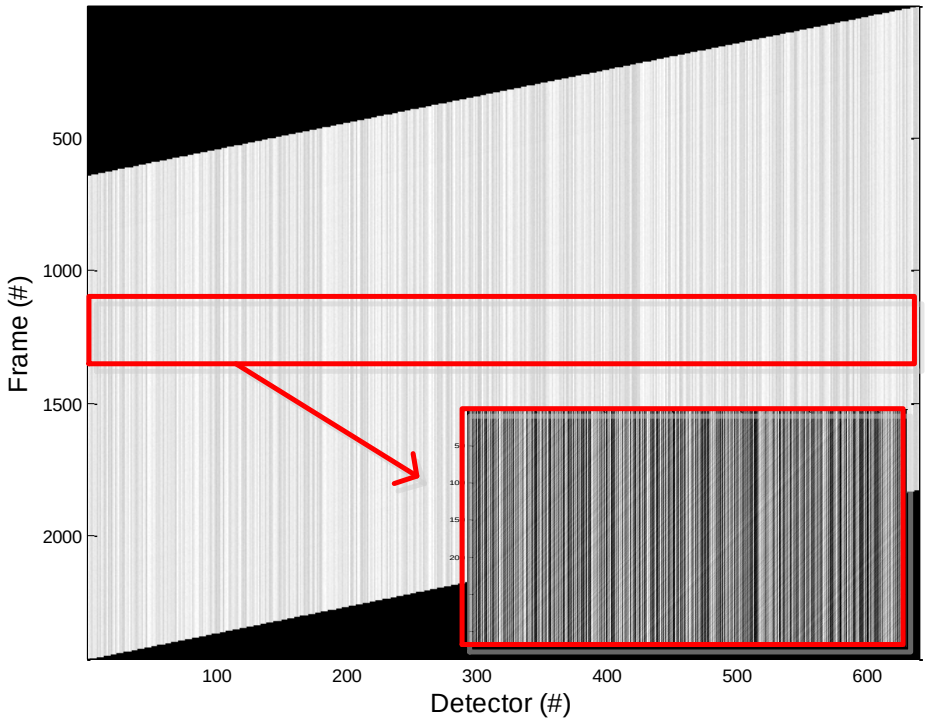


Figure 2: Sample of a TIRS Yaw Scene (Simulated).

2. Shift all pixels so corresponding samples line up. Because of the yaw maneuver and the layout of the focal plane, all detectors will see the same spot on the ground with a one pixel lag for TIRS and a two pixel lag for OLI. Because of this pixel lag all samples need to be shifted by increasing multiples of one or two. Figures 1 and 2 show how the pixels need to be shifted for TIRS. Whether the pixels are shifted up or down depends on the orientation of the instrument. (At this time it is thought that the instrument will be yawed at -90^{o})

    a. As a result of shifting, the frames at the top and bottom will contain invalid image data. Any frame which does not contain all image data is considered excess here. All excess

pixels need to be removed from the analysis. Figure 3 shows these excess pixels. If there are 640 detectors in a TIRS SCA there will be 640 excess frames on the top and bottom of the image after shifting.

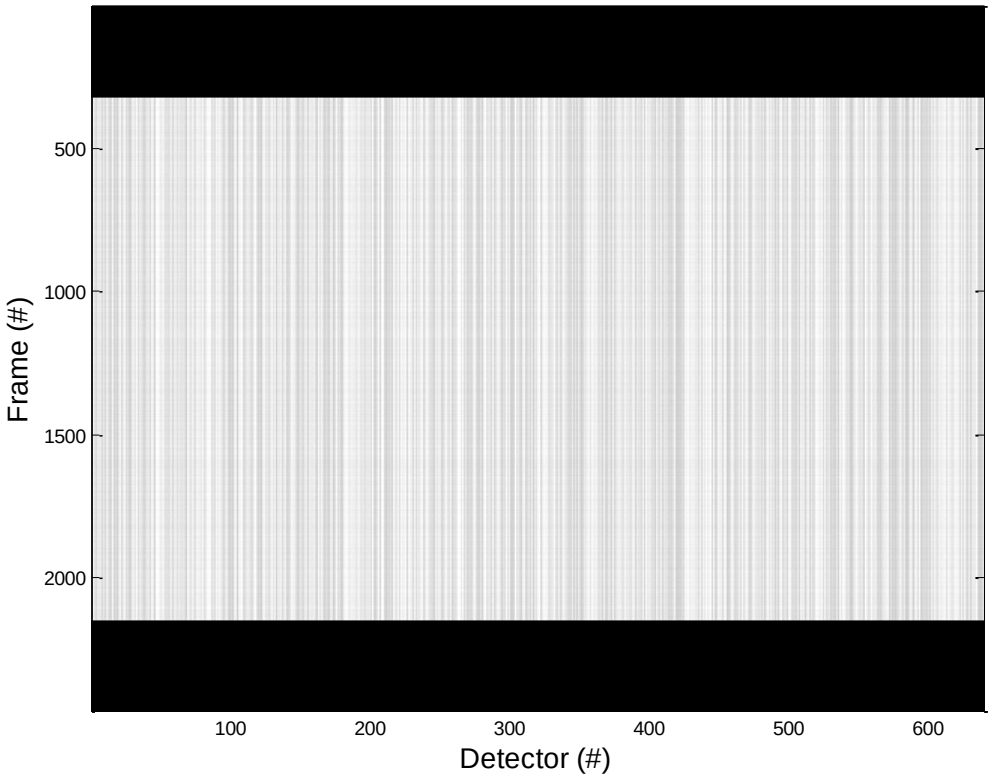


Figure 3: Sample of a TIRS Yaw Scene (Detectors Shifted with Added Fill Data).

3. Run the SMA algorithm on the detector samples to find the detector relative gain estimations.

   SMA 2

   a. First square all pixel values. (Samples from the last detector do not need to be considered).

   $$Q_{ij}^2$$

   where *Q* is the pixel value in DN, *i* is the row, and *j* is the column.

   b. Then multiply all adjacent pixels. (The maximum value of *j* here is one less than the number of detectors.)

   $$Q_{ij}Q_{ij+1}$$

   c. Both of these products are summed all each column.

   $$\sum_i Q_{ij}^2 \quad , \quad \sum_i Q_{ij}Q_{ij+1}$$

d. We now should have a summed square product and summed cross product for each detector except for the last. These summations are put into a matrix with the adjacent pixel multiplication product set negative.

$$\begin{bmatrix} \sum_i Q_{i1}^{o^2} & -\sum_i Q_{i1}^o Q_{i2}^o & 0 & 0 & 0 & \cdots & 0 \\ -\sum_i Q_{i2}^o Q_{i1}^o & 2\sum_i Q_{i2}^{o^2} & -\sum_i Q_{i2}^o Q_{i3}^o & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & -\sum_i Q_{im-1}^o Q_{im-2}^o & 2\sum_i Q_{im-1}^{o^2} & -\sum_i Q_{im-1}^o Q_{im}^o \\ 0 & \cdots & 0 & 0 & 0 & -\sum_i Q_{im}^o Q_{im-1}^o & \sum_i Q_{im}^{o^2} \end{bmatrix}$$

where *m* is the number of detectors

e. The reciprocal relative gain matrix and the zero matrix are added to the summation matrix again, so the relative gains can be solved for. The zero matrix here is not actually zero but the smallest possible positive non zero number the computer can use.

$$\begin{bmatrix} \sum_i Q_{i1}^{o^2} & -\sum_i Q_{i1}^o Q_{i2}^o & 0 & 0 & 0 & \cdots & 0 \\ -\sum_i Q_{i2}^o Q_{i1}^o & 2\sum_i Q_{i2}^{o^2} & -\sum_i Q_{i2}^o Q_{i3}^o & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & -\sum_i Q_{im-1}^o Q_{im-2}^o & 2\sum_i Q_{im-1}^{o^2} & -\sum_i Q_{im-1}^o Q_{im}^o \\ 0 & \cdots & 0 & 0 & 0 & -\sum_i Q_{im}^o Q_{im-1}^o & \sum_i Q_{im}^{o^2} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{m-1} \\ r_m \end{bmatrix} = \begin{bmatrix} \approx 0 \\ \approx 0 \\ \vdots \\ \approx 0 \\ \approx 0 \end{bmatrix}$$

f. The reciprocal of the reciprocal relative gains will have to be taken in order to obtain the relative gains.

g. After the relative gains have been found, they will have to be normalized to one.

4. Run the histogram algorithm on the detector samples to find the classic detector relative gain estimations.

   The detector response is assumed to follow the linear model:

$$Q_{i,j} = g_{i,j} \cdot L_\lambda + b_{i,j}$$

   After subtraction of the detector bias to obtain $Q_{i,j}'$ The relative gain of the detector would be:

$$rg_{i,j} = \frac{g_{i,j}}{\bar{g}} = \frac{Q'_{i,J}}{\bar{Q}}$$

5. Repeat steps 2 to 4 for all SCAs and bands.

7.4.16 SCA Overlap Statistics Characterization

7.4.16.1 Background

Each SCA as collected exhibits a different overall brightness when compared to other SCAs. This algorithm will derive SCA to SCA ratios that will be used to normalize the gain of all the SCAs. Once these ratios are used to correct the discontinuities, all the SCAs should exhibit the same relative

brightness for a uniform input radiance field. Ratios are calculated on a per-scene basis, and then stored in the trending database.

7.4.16.2 Input

Descriptions	Symbol	Units	Level	Source	Type
Scene			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float
Nominal Detector Shifts			N_{bands} x N_{SCAs}	CPF	Int
Valid Correlation Shift Maximum			N_{bands}	CPF	Int
Minimum Valid Neighboring Segments			N_{bands}	CPF	Int
Artifact Mask			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Saturated Pixel List			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Impulse Noise			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Dropped Frames			N_{bands} x N_{SCAs}	LM	Int
Inoperable Detector List			N_{bands} x N_{SCAs}	CPF	Int
Systematic Model			N_{bands} x N_{SCAs} x 2	CPF	Int
Nominal Scene Elevation*	d	km	1	DEM	Float
SCA Offsets	O_{SCA}	Pixels	N_{bands} x N_{SCAs}	CPF	Int
SCA_Overlap_Threshold	STR_{band}		N_{band}	CPF	Float
Default_SCA-SCA ratios			N_{band} x N_{SCA}	CPF	Float

*The nominal scene elevation can be the average of the minimum and maximum elevation of the scene.

7.4.16.3 Output

Descriptions	Symbol	Units	Level	Target	Type
SCA-SCA ratios			N_{bands} x N_{SCAs}	Db	Float
Overlap Pixel Standard Deviations			N_{bands} x N_{SCAs}	Db	Float
Segment Percentage			N_{bands} x N_{SCAs}	Db	Float
Invalid SCA Correlation Flag			N_{bands} x N_{SCAs}-1	Db	Int
SCA Overlap Threshold Flag			N_{band}	Db	Int

7.4.16.4 Options

Write ratios, standard deviations, and percentages to database (On by Default)
Summary Report (Off by Default). Additionally, the option to use just the SCA offsets from the CPF (O_{SCA}), without doing the extra calculations, enables non-Earth scenes to be aligned manually.

7.4.16.5 Procedure

1. Read in the processing parameters.
2. Read in SCA #1 and #2.
3. Using the systematic model, with nominal scene elevation, compute the along- and across-track offsets to align the SCAs. This nominal SCA alignment will be similar to the "left Overlap" method described in the L1R SCA Stitching algorithm; the difference is using the model instead of the Legendre coefficients for alignment. The along-track offset is how many lines to add to the trailing SCA to align it with the leading SCA. The across-track offset is the number of samples of overlap. These two values will be added to the CPF values for SCA overlap and offset (default zero) to allow the user to adjust the overall alignment on a per-scene basis.
    a. Calculate the geodetic latitudes and longitudes of the trailing (search) SCA on the first line of data. Only a subset of the line of samples need to be calculated, just enough to extend beyond the expected overlap.
    b. Calculate the geodetic latitude and longitude of either the first or last sample of the leading (reference) SCA line index 0.
    c. Find the sample in the search SCA with the minimum distance to the reference sample. Save the distance and search sample index and reference line index for the next iteration.
    d. Repeat b, incrementing the line index and step c. until the saved distance is smaller than the minimum distance found.
    e. Calculate the along- and across-track offsets based on the line and sample indices of the two pixels with minimum distance found in step c.

4. Calculate cross-track and along-track correlations.
    a. Match the detector responses of the overlap region (The overlap region is described in Figure 1). This is basically calculating and applying relative gains, but the gains are only derived from the current image, and only the detectors in the overlap region are used. Relative gains have already been applied, but depending on the scenes adjacent

detectors may still not be perfectly matched. The stripes caused by the lack of matching causes problems with correlating the two SCAs. So the overlap detectors must be matched here to ensure there aren't any stripes in the overlap region.

$$x'_{m,n} = x_{m,n} \cdot \frac{\dfrac{\overline{x_1} + \overline{x_2} + \overline{x_3} + \overline{x_4} + \overline{x_5} + \cdots + \overline{x_L}}{\#\,\text{Operable Detectors in OVLap Region}}}{\overline{x_n}}$$

Where x' is a pixel in the overlap region after it has been matched, x is a pixel in the overlap region before being matched, *m* is a row (frame), and *n* is a column (detector). $\overline{x_1}$ is the mean of the first detector in the overlap region, $\overline{x_L}$ is the mean of the last detector in the overlap region, and $\overline{x_n}$ is the mean of the detector containing the $x_{m,n}$ pixel.

Pixels in the artifact mask and samples from inoperable detectors should be ignored when they are encountered. (If $x_{m,n}$ is a pixel in the artifact mask or a sample from an inoperable detector it should not be changed, $x'_{m,n} = x_{m,n}$). Pixels in the artifact mask and samples from inoperable detectors should also not be used in the calculation of the detector means.

As shown in Figure 1, the overlap region is defined as the region of image data between adjacent SCAs that is 18 + # of overlap detectors wide (9 adjacent detectors in SCA #1 + overlap detectors + 9 adjacent detectors in SCA #2) by # of frames common to both SCAs after nominal detector shift alignments.

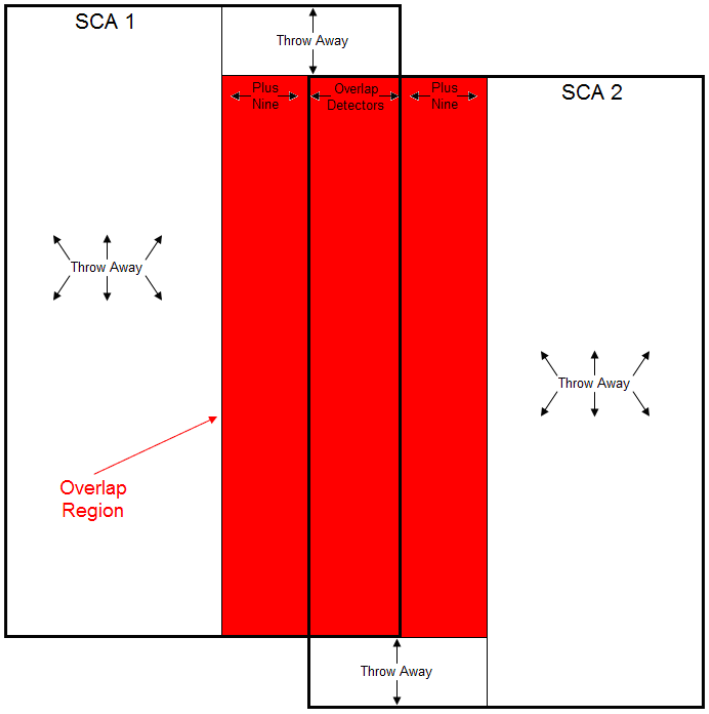


Figure 1: Overlap Region.

b. After SCA 1's overlap region detectors are matched, the overlap region should be broken up into 101 frame segments taken every 50 frames. (This means the first pixel in every segment is the middle pixel from the previous segment, and the last pixel in every

segment is the center pixel of the next segment. So every segment overlaps with 50% of its neighbor segments.) A 10 frame buffer should be left at the top of the overlap region, so the first segment will start at the 11th frame of the overlap region. At least a 10 frame buffer should also be left at the bottom of the image. This buffer could be larger if there are any remainder frames left from the segmenting process.

c. After SCA 2's overlap region is matched, it will be divided up into 119 frame segments with centers "equal" to the left SCA segment centers. The total number of segments from each SCA will be equal to the number of complete segments that can be obtained from the right SCA or

$$\#\text{segments} = floor\left(\frac{\#\text{frames - nominal detector shift - 119}}{50}\right).$$

Figure 2, below, shows how each SCA should be segmented. All dimensions are inclusive (contains both top and bottom frames) so each alternate"101" segment contains the frame between them. The dotted lines do not indicate the center of the segments, but rather the start and end of segments. In SCA 1 the centers of segments are equal to the first frames of other segments within SCA 1, but in SCA 2 the centers do not match up with the first frames of other segments with in SCA 2.

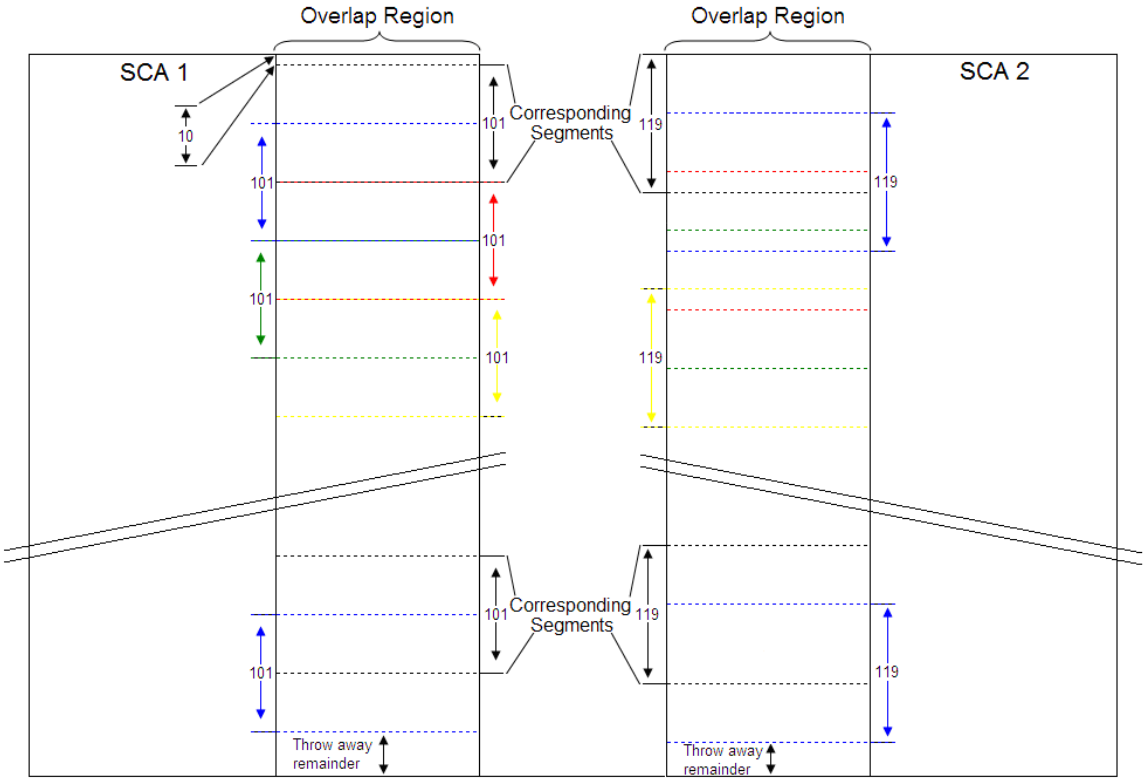


Figure 2: SCA Segmentation.

d. Once the two overlap regions are segmented, we will find horizontal and vertical differences within each segment. (The horizontal difference is the right pixel minus left pixel. We do this for every pixel in the segment. The resultant difference matrix will have one less column than the segment. The vertical difference is the same but with the bottom pixel minus the top pixel. So the dimensions of these difference matrices will be the #overlap detectors +8 x 100 for left SCAs and #overlap detectors + 8 x 118 for the right SCAs). These horizontal and vertical differences are explained more below. Every time a pixel in the artifact mask or a sample from an inoperable detector is encountered, the difference should be set to zero. (So every pixel in the artifact mask will produce 4

zero entries in the difference matrix, and inoperable detectors will cause two columns to contain all zeros).

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{3,5} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} \\ x_{5,1} & x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} \end{bmatrix} \rightarrow \begin{bmatrix} x_{1,2}-x_{1,1} & x_{1,3}-x_{1,2} & x_{1,4}-x_{1,3} & x_{1,5}-x_{1,4} \\ x_{2,2}-x_{2,1} & x_{2,3}-x_{2,2} & x_{2,4}-x_{2,3} & x_{2,5}-x_{2,4} \\ x_{3,2}-x_{3,1} & x_{3,3}-x_{3,2} & x_{3,4}-x_{3,3} & x_{3,5}-x_{3,4} \\ x_{4,2}-x_{4,1} & x_{4,3}-x_{4,2} & x_{4,4}-x_{4,3} & x_{4,5}-x_{4,4} \\ x_{5,2}-x_{5,1} & x_{5,3}-x_{5,2} & x_{5,4}-x_{5,3} & x_{5,5}-x_{5,4} \end{bmatrix} \rightarrow$$

Example Image Segment — Horizontal Difference

$$\begin{bmatrix} ((x_{2,2}-x_{2,1})-(x_{1,2}-x_{1,1})) & ((x_{2,3}-x_{2,2})-(x_{1,3}-x_{1,2})) & ((x_{2,4}-x_{2,3})-(x_{1,4}-x_{1,3})) & ((x_{2,5}-x_{2,4})-(x_{1,5}-x_{1,4})) \\ ((x_{3,2}-x_{3,1})-(x_{2,2}-x_{2,1})) & ((x_{3,3}-x_{3,2})-(x_{2,3}-x_{2,2})) & ((x_{3,4}-x_{3,3})-(x_{2,4}-x_{2,3})) & ((x_{3,5}-x_{3,4})-(x_{2,5}-x_{2,4})) \\ ((x_{4,2}-x_{4,1})-(x_{3,2}-x_{3,1})) & ((x_{4,3}-x_{4,2})-(x_{3,3}-x_{3,2})) & ((x_{4,4}-x_{4,3})-(x_{3,4}-x_{3,3})) & ((x_{4,5}-x_{4,4})-(x_{3,5}-x_{3,4})) \\ ((x_{5,2}-x_{5,2})-(x_{4,2}-x_{4,1})) & ((x_{5,3}-x_{5,2})-(x_{4,3}-x_{4,2})) & ((x_{5,4}-x_{5,3})-(x_{4,4}-x_{4,3})) & ((x_{5,5}-x_{5,4})-(x_{4,5}-x_{4,4})) \end{bmatrix}$$ Vertical Difference

(Difference Matrix)

x represents a pixel.

e. Now correlate each difference matrix from SCA 1 with each difference matrix from SCA 2. Each difference matrix from SCA 1 is only correlated with the one corresponding difference matrix from SCA 2.

   i. Start with the last column from SCA 1's difference matrix overlapping the first column of SCA 2's difference matrix, and the first row of SCA 1's difference matrix overlapping the first row of SCA 2's difference matrix. There will only be 100 "pixels" (these "pixels" are actually the four pixel difference elements from the difference matrix) overlapping at this point. This corresponds to a shift of -8 horizontal and -9 vertical. To find the correlation product for this shift, multiply all the overlapping "pixels" and find the mean of their products. Figure 3 shows the overlapping "pixels" of two example difference matrices.

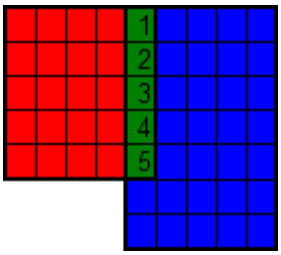


Figure 3: Two Overlapping Difference Segments (Red = SCA 1 Diff. Mat., Blue = SCA 2 Diff. Mat., and Green = Overlapping "Pixels").

The correlation product of the above figure is the mean of the product of the 1st overlapping "pixel" pair, the product of the 2nd overlapping "pixel" pair, the product of the 3rd overlapping "pixel" pair, the product of the 4th overlapping "pixel" pair, and the product of the 5th overlapping "pixel" pair.

   ii. Shift the two difference matrices, and find the correlation products by finding the mean of the individual overlapping "pixel" products. Continue doing this until all columns of both difference matrices are overlapping, and the bottom frames of each matrix are overlapping. (This corresponds to a shift of 9 horizontal and

9 vertical). Figure 4 shows the two example difference matrices at different shifts. The red or the matrix from SCA 1 can not be seen in the images on the right side since it is completely overlapped with the blue or SCA 2's matrix.

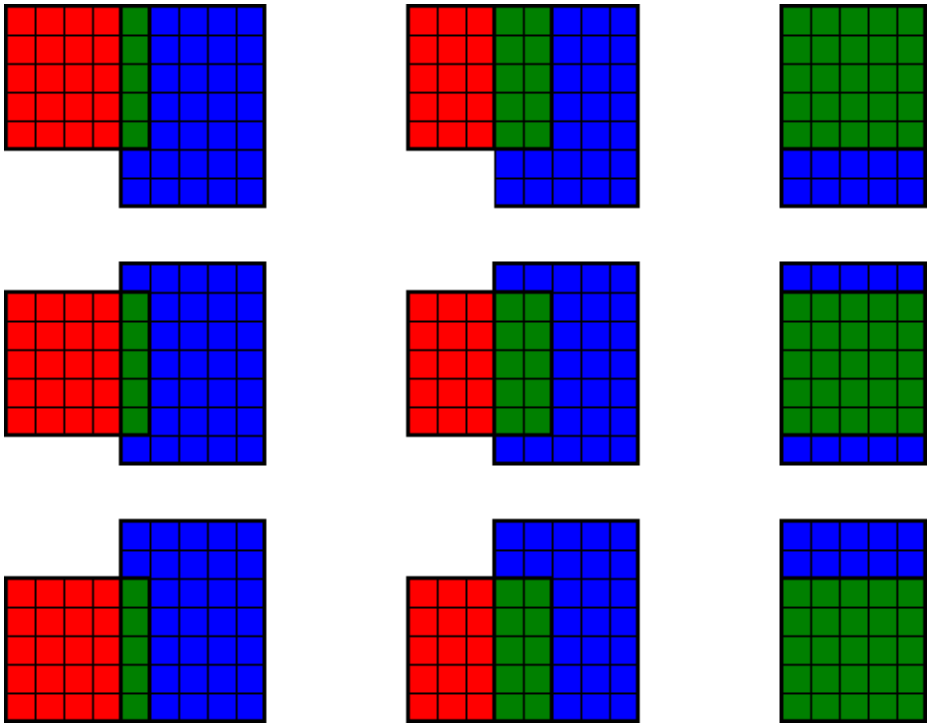


Figure 4: Two Overlapping Difference Matrices.

Each of the different shifts seen above will have a different correlation product. In this example there should be 20 different correlation products. In the actual scene there will be 342 different correlation products between two difference matrices as the shifts range from -8 horizontal and -9 vertical and 9 horizontal and 9 vertical.

One must be careful when calculating the mean of the overlapping "pixels". If the product between two "pixels" is zero, that pair must be removed from calculating the mean.

iii. After all 342 correlation products have been found for the two difference matrices, the correlation shifts should be found for the maximum correlation product. Figure 5 is an example of the correlation products from one set of difference matrices. The maximum correlation product is represented by the brightest white pixel and corresponds to a shift of 0 horizontal and 1 vertical.

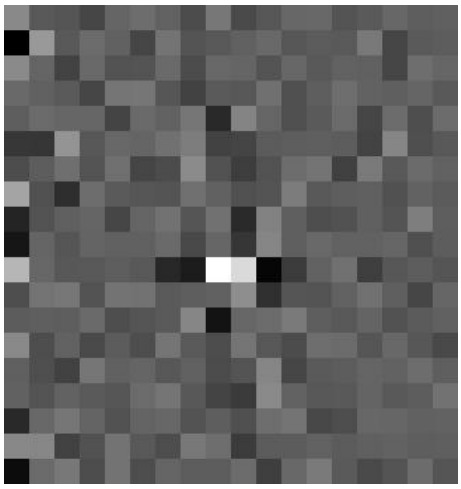


Figure 5: Correlation Products.

iv. Steps 4.e.i-iii should be run for each set of difference matrices.

v. Now we have a horizontal and vertical correlation shift for all the difference matrices. If either the absolute vertical or horizontal shift is larger than the Valid Correlation Maximum, that correlation will not be considered valid. Also a correlation, both horizontal and vertical shifts, must be equal to the correlation shifts of its neighboring difference matrices to be valid. The number of neighbor shifts it must equal is set with the Minimum Valid Neighboring Segments variable. If the Minimum Valid Neighboring Segments is even, the current correlation shifts are compared to the same number of higher and lower correlation shifts. (Since the image is segmented from top to bottom, higher and lower correlation shifts denote the correlation shifts belonging to the adjacent difference matrices on the top and bottom of the current difference matrix.) If the Minimum Valid Neighboring Segments is odd, the current correlation shifts are compared to one more higher correlation shift than lower correlation shift. So if the Minimum Valid Neighboring Segments is set to 1, the current correlation product is compared to only its adjacent higher correlation shift. The edge correlation products should only be compared to the same number of higher and lower correlation shifts as the central correlation shifts. If the Minimum Valid Neighboring Segments is set to 2, all the central correlation shifts are compared to one higher and one lower correlation shift, and the edge correlation products are only compared to one either higher or lower correlation shift. Figures 6 and 7 show some example correlation shifts (Minimum Valid Neighboring Segments is set to 2, and the Valid Correlation Maximum is set to 6).

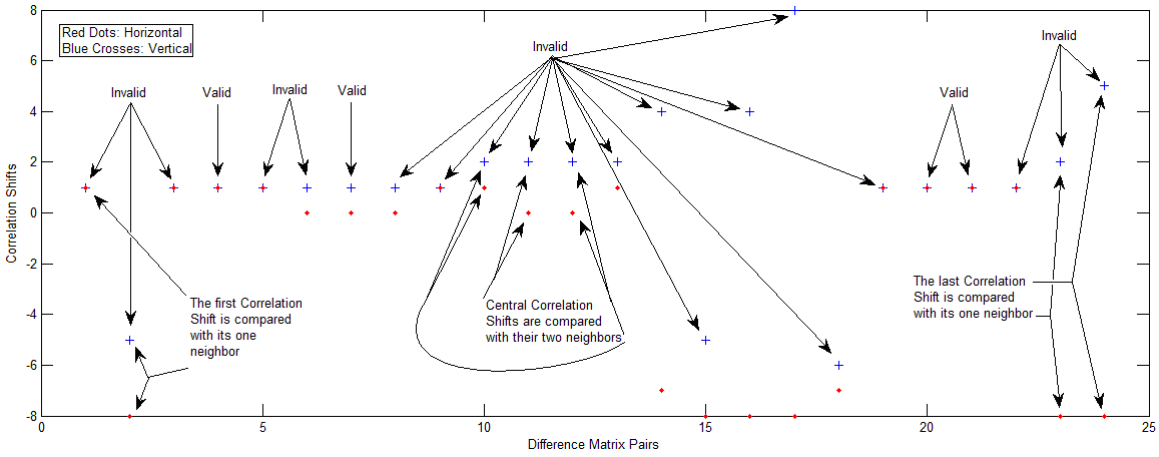


Figure 6: Correlation Shifts.

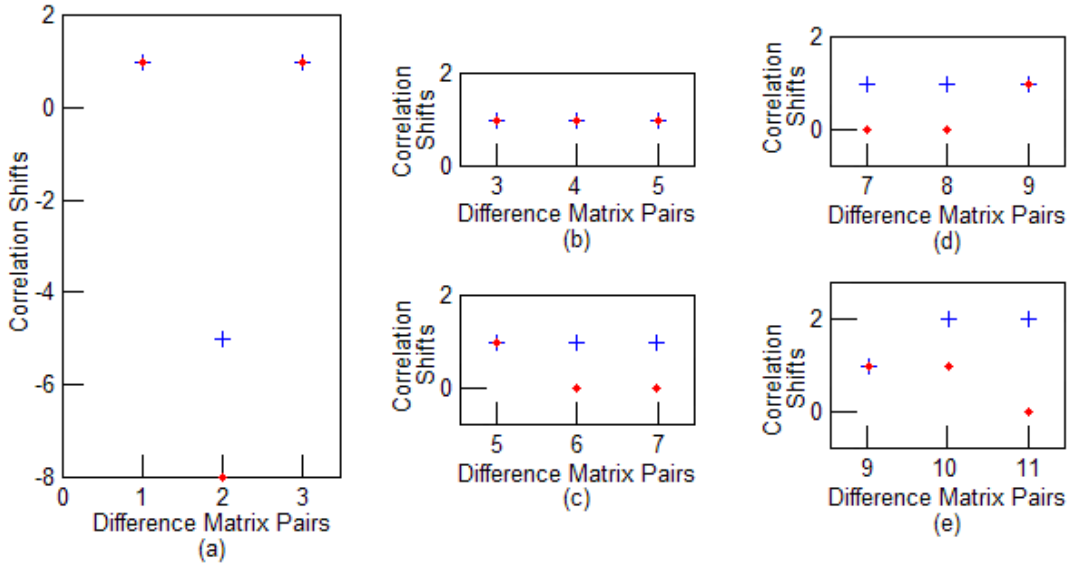


Figure 7: Close-up of Correlation Shifts.

In Figure 7 (a), all of the correlation shifts are invalid. The 1st shifts are invalid because neither the horizontal (Red Dot) nor the vertical shifts (Blue Cross) are the same as the 2nd shifts. The 2nd shifts are invalid because both the horizontal and vertical shifts are different from the 1st shifts; these shifts are also invalid because they are different from the 3rd shifts, and the absolute horizontal shift is greater than 6. The 3rd shifts are invalid because both the horizontal and vertical shifts are different from the 2nd shifts. The 4th set of shifts is valid because both the horizontal and vertical shifts are the same as the 3rd and 5th, and both the absolute horizontal and vertical shifts are less the Valid Correlation Maximum. The 5th shifts are invalid because the horizontal shift is different from the horizontal shift of the 6th set of shifts. The 6th shifts are invalid because they are not equal to the 5th. The 7th set of shifts is valid because both shifts are equal to both the 6th and 8th shifts, and they are less then 6. The 8th, 9th, 10th, and 11th are all invalid because their shifts are not equal to both of their neighbor's shifts.

5. Once all the valid correlation shifts have been found, those shifts are used to line up the corresponding segments. Individual pixel-pixel ratios can now be calculated by taking each overlapping pixel in SCA 1 and dividing it by each overlapping pixel in SCA 2. **If there were not any valid correlated segments, we cannot find an SCA-SCA ratio, and the Invalid SCA Correlation Flag needs to be set, and steps 7-9 can be skipped.**

6. If the SCA-SCA ratio exceeds the SCA Overlap Threshold for that band, then the SCA Overlap Threshold Flag needs to be set, and the SCA ratio should not be updated. The SCA Threshold Flag is a value around the mean Default SCA-SCA ratio.
7. Calculate the mean of all the overlapping pixel ratios; this is the SCA-SCA ratio.
8. Calculate the standard deviation of all the overlapping pixel ratios.
9. Calculate the percentage of valid segments of the total segments.
10. Keep SCA 2 and read in SCA 3.
11. Repeat steps 2-10 for all Bands and SCAs.
12. If the writing to database option was selected, write the SCA-SCA ratios, standard deviations, and percentages to the database. (These are not updated if the SCA Overlap Threshold Flag is set). If the SCA Correlation Flag was set for this SCA-SCA pair, the SCA-SCA ratio and standard deviation should be set to some value to indicate this. (-1?)
13. If the summary report option was selected, write the SCA-SCA ratios, standard deviation, and percentages to a report. The report will include a line for the SCA Overlap Threshold Flag that has a 1 if the STR is violated and a 0 otherwise.

7.4.17 SCA Discontinuity Correction

7.4.17.1 Background

Image data obtained from the EO-1 Advanced Land Imager (ALI) used to simulate expected OLI data sets exhibited significant residual “discontinuities” in overall response across SCAs after relative gain correction and absolute radiometric correction. It is anticipated that actual OLI and TIRS data will have the same issues; hence a need for the correction algorithm discussed herein is anticipated. The SCA overlap statistics algorithm calculates SCA-SCA ratios which measure the difference in brightness between two adjacent chip assemblies. The characterization algorithm uses these ratios to calculate SCA discontinuity correction factors, which are applied in the correction algorithm, thus creating a more uniform cross track data product.

7.4.17.2 Input

Descriptions	Symbol	Units	Level	Source	Type
Scene	Q	$W \cdot sr^{-1} \cdot m^{-2} \cdot \mu m$)	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float
SCA-SCA Ratios					
a) CPF Model		Unitless	N_{bands}	CPF	Float
b)Scene Specific		Unitless	N_{bands}	Db	Float
Saturated Pixel List			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Impulse Noise			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Dropped Frames			N_{bands} x N_{SCAs}	LM	Int

Inoperable Detectors			N_{bands} x N_{SCAs}	CPF	Int

7.4.17.3 Output

Descriptions	Symbol	Units	Level	Destination	Type
Scene		$W \cdot sr^{-1} \cdot m^{-2} \cdot \mu m)$	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float

7.4.17.4 Options

SCA Discontinuity-ratios

a) CPF Model (Selected if any database "Invalid SCA Correlation Flags" are found set by the SCA Overlap Statisics Characterization algorithm for any SCA pair in the current scene band), CPF values will initially be set equal 1, effectively disabling a correction to the scene, yet allowing for a future "static" correction if a suitable set of CPF correction parameters can be determined. This option should also be selectable during work order creation regardless of flag states. (Default-Off)
b) Scene Specific SCA-SCA Ratios (Default-On)

Summary Report (Default-Off)

7.4.17.5 Procedure

14. Read in the processing parameters.
15. For each band do the following:
    a. Check if any Invalid SCA Correlation Flags are set, and if not then execute using the SCA-SCA ratios. If any flags are set, then execute using the CPF correction parameters.
    b. Set the leftmost (westernmost) SCA discontinuity correction factor to 1. For all other SCAs, set the discontinuity correction factor to its SCA-SCA ratio and propagate these ratios to the right. (This means the discontinuity correction factor for the second SCA is equal to its SCA-SCA ratio. The discontinuity correction factor of the third SCA is equal to its SCA-SCA ratio multiplied by the discontinuity correction factor of the second SCA. The discontinuity correction factor for any SCA is its own SCA-SCA ratio multiplied by the discontinuity correction factor of the SCA to the west or left)
    c. Normalize all the SCAs' discontinuity correction factors to 1. (This is done on the correction factors and not the ratios. Simply add up all the discontinuity correction factors, including the 1 used for the westernmost SCA, and divide by the amount of SCAs in the band. Take the reciprocal of this division, and multiply it with all the discontinuity correction factors).
    d. Multiply each pixel, excluding any problem pixels from the Artifact Mask, in an SCA by its discontinuity correction factor.
    e. If the summary report option was selected, write the discontinuity correction factors to a summary report.

7.4.18 Inoperable Detectors Fill

7.4.18.1 Background

An inoperable detector is one that provides no change in output DN value when radiance on its input is changed. The data from inoperable detectors appear as very distinct stripes in final image products. This algorithm replaces image data generated by the inoperable detectors with the data from neighboring detectors, in order to enhance visual appearance of OLI and TIRS images. It does not replace data generated by the "out-of-spec" detectors (TIRS may have different way of dealing with "out-of-spec" detectors). The algorithm assumes that the input scene data are nominally spatially aligned and that a list of known inoperable detectors is available within the CPF. The algorithm operates within the normal Level 1R data processing flow.

7.4.18.2 Inputs

Descriptions	Symbol	Units	Level	Source	Type
Scene (L1R) Earth data	*L*	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float
Inoperable detectors			N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Int

7.4.18.3 Outputs

Descriptions	Symbol	Units	Level	Target	Type
Scene (L1R-corrected)	*L*	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float

7.4.18.4 Options

None.

7.4.18.5 Procedure

For each inoperable detector, d_i, listed in the CPF

1. If the inoperable detector's immediate neighbors are operable, replace each pixel in the image column generated by the inoperable detector with the mean of two adjacent pixels in the same line, *l*, of the same SCA:

$$Q(l,d_i) = \frac{Q(l,d_{i-1}) + Q(l,d_{i+1})}{2}$$

- If the inoperable detector is the first (d_1) or last (d_{max}) on an SCA, its data will be replaced with the data from the nearest neighboring operable detector from the same SCA.

$$Q(l,d_1) = Q(l,d_2)$$

$$Q(l, d_{\max}) = Q(l, d_{\max-1})$$

2. If there are two consecutive inoperable detectors, d_i and d_{i+1}, on the same SCA, then the correction of corresponding image data needs to be accomplished by interpolating data from the available adjacent operable detectors:

$$Q(l, d_i) = \frac{2 \times Q(l, d_{i-1}) + Q(l, d_{i+2})}{3}$$

$$Q(l, d_{i+1}) = \frac{Q(l, d_{i-1}) + 2 \times Q(l, d_{i+2})}{3}$$

For example, if detectors 23 and 24 are inoperable, then their outputs need to be replaced with the result of linear interpolation of data from detectors 22 and 25, for each line of L1R image.

- If the two inoperable detectors are the first and second detector on an SCA, their data will be replaced with the data from the third detector from the same SCA.

$$Q(l, d_1) = Q(l, d_3)$$
$$Q(l, d_2) = Q(l, d_3)$$

- If the two inoperable detectors are the last and second last detectors on an SCA, their data will be replaced with the data from the third last detector from the same SCA.

$$Q(l, d_{\max}) = Q(l, d_{\max-2})$$

$$Q(l, d_{\max-1}) = Q(l, d_{\max-2})$$

3. If there are more than two consecutive inoperable detectors on an SCA, then the pixel values for the affected detectors need to be filled with zeros.

Note: To replace inoperable detector data, this algorithm uses pixels affected by artifacts, e.g. impulse noise or saturation, and fill pixels (on top and bottom of images that support the nominal image alignment) the same way as regular pixels.

7.4.19 Residual Striping Correction

7.4.19.1 Background

Many algorithms exist to estimate relative detector gain. These algorithms can generate estimates that do a good job of striping correction in typical scenes most of the time. However, scenes can still contain observable striping after radiometric correction (including relative gain correction). The most likely causes include:

- underestimation or overestimation of detector bias levels, leading to errors in bias removal
- underestimation or overestimation of relative gains, due to algorithmic issues and/or potential issues relating to scene content

- underestimation or overestimation of absolute gains at the band and/or focal plane module level

Correction of residual striping in a scene may be desirable for cosmetic reasons and/or required for some analytical studies.

The algorithm described herein implements a cosmetic approach for residual striping removal in radiometrically corrected image data assuming a standard relative gain correction has already been applied. Data obtained from the striping characterization algorithm will be used to drive the Residual Striping Correction algorithm.

7.4.19.2 Inputs

Descriptions	Symbol	Units	Level	Source	Type
Scene			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float
Scene Striping Correction Matrix			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Striping Characterization Algorithm	Float

7.4.19.3 Outputs

Descriptions	Symbol	Units	Level	Destination	Type
Scene			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float

7.4.19.4 Options

No Options

7.4.19.5 Procedure

1. Read in the image.
2. Read in the Scene Striping Correction Matrix from the Striping Characterization Algorithm.
3. Subtract the Striping Correction Matrix from the image. For each FPM, the Striping Correction Matrix will have one less pixel on all sides (not including fill data, so the first and last columns and rows should be unchanged. All artifacts and inoperable detectors were zeroed out in the Striping Characterization algorithm; hence no special handling is required in this algorithm. The Striping Correction Matrix simply needs to be aligned with the image data and subtracted.

7.4.20 Saturated Pixel Replacement

7.4.20.1 Background

The saturated pixels determined in L0R data using the Saturated Pixel Characterization algorithm may change their values during radiometric processing. As a result, they may not be easily identified in final L1R product. To avoid erroneous interpretation of radiometric data in L1R products, it is important to clearly locate originally identified saturated pixels.

This algorithm describes the post-1R correction to replace high-end saturated pixels in image data with the band-maximum radiance (L_{sat_max}) values and low-end saturated pixels with the band-minimum radiance (L_{sat_min}) values. The algorithm operates within the normal Level 1R data processing flow.

7.4.20.2 Inputs

Descriptions	Symbol	Units	Level	Source	Type
Scene (L1R) Earth data	L	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float
Saturated pixel locations			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Int
Low Saturation Radiance Level	L_{sat_min}	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands}	CPF	Float
High Saturation Radiance Level	L_{sat_max}	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands}	CPF	Float

7.4.20.3 Outputs

Descriptions	Symbol	Units	Level	Target	Type
Scene (L1R-corrected)	L	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float

7.4.20.4 Options

By default, the digitally low and high saturated pixels will be replaced with the corresponding low and high saturation radiance levels. In addition, the following options need to be selectable through work order parameters:

- replace pixels only at the high saturation level or only at the low saturation level
- replace both analog and digitally saturated pixels (with same radiance saturation levels).
- no saturated pixel replacement.

7.4.20.5 Procedure

For each detector (d):

1. Based on the selected options and Labeled Mask record, find each pixel originally identified in Saturated Pixel Characterization algorithm as low and/or high saturated

2. Replace the pixel values with the corresponding Low and High Saturation Radiance Levels, given in CPF

7.4.21 Radiance Rescaling

7.4.21.1 Background

The standard LDCM products are OLI reflectance and TIRS radiance products in 16-bit integer format. The OLI and TIRS data are radiometrically and geometrically processed using floating point operations. For OLI, this algorithm scales and converts the resultant L1G reflectance from floating point format to 16-bit integer format using scaling parameters from the CPF. In addition, the algorithm provides rescaling coefficients for direct conversion from 16-bit integer reflectance to floating point radiance. For TIRS, the algorithm scales and converts radiance values from floating point format to 16-bit integer format. The algorithm assumes that no scaling is applied during the Geometric Processing.

7.4.21.2 Inputs

Descriptions	Symbol	Units	Level	Source	Type
For OLI:					
Reflectance scene (L1G)	ρ		N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Geometric Processing	float
Reflectance Rescaling Gain	G_ρ	DN^{-1}	N_{bands}	CPF	float
Reflectance Rescaling Bias	B_ρ		N_{bands}	CPF	float
Reflectance to Radiance conversion coefficient	ρ_R	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands}	Reflectance Conversion	float
For TIRS:					
Radiance scene (L1G)	L	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Geometric Processing	float
Radiance Rescaling Gain	G_L	$\frac{W}{m^2 \cdot sr \cdot \mu m} / DN$	N_{bands}	CPF	float
Radiance Rescaling Bias	B_L	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands}	CPF	float

7.4.21.3 Outputs

Descriptions	Symbol	Units	Level	Target	Type
For OLI:					
Scene L1G	ρ_{int}	DN	N_{bands} x N_{SCAs} x $N_{detectors}$		Int (16-bit)

			x N_{frames}		
Reflectance Rescaling Gain	G_ρ	DN^{-1}	N_{bands}	Metadata	float
Reflectance Rescaling Bias	B_ρ		N_{bands}	Metadata	float
Radiance Rescaling Gain	G_L	$\frac{W}{m^2 \cdot sr \cdot \mu m} / DN$	N_{bands}	Metadata	float
Radiance Rescaling Bias	B_L	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands}	Metadata	float
For TIRS:					
Scene (L1G)	L_{int}	DN	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Int (16-bit)
Radiance Rescaling Gain	G_L	$\frac{W}{m^2 \cdot sr \cdot \mu m} / DN$	N_{bands}	Metadata	float
Radiance Rescaling Bias	B_L	$\frac{W}{m^2 \cdot sr \cdot \mu m}$	N_{bands}	Metadata	float

7.4.21.4 Options

7.4.21.5 Procedure

1. For each band, apply scaling to each image pixel, except to fill data, of geometrically corrected (L1G) floating point OLI reflectance and TIRS radiance images:

   a. For OLI

$$\rho_{scal} = \frac{\rho - B_\rho}{G_\rho}$$

   b. For TIRS

$$L_{scal} = \frac{L - B_L}{G_L}$$

2. Convert scaled OLI reflectance (ρ_{scal}), TIRS radiance (L_{scal}) and fill data pixel values from floating point to 16-bit integer format through rounding to the closest 16-bit integer values

   a. For OLI

$$\rho_{\text{int}} = round\left(\rho_{scal}\right)$$

   b. For TIRS

$$L_{\text{int}} = round\left(L_{scal}\right)$$

3. Under the assumption that 0 will be reserved for fill data, convert all zeros in ρ_{int} and L_{int} images to Q_{calmin} = 1. More generally, convert all pixels with value less than Q_{calmin} to Q_{calmin}.

4. Only for OLI, calculate rescaling coefficients that will be used for direct conversion from 16-bit integer reflectance to floating point radiance

$$G_L = \rho_R \cdot G_\rho$$
$$B_L = \rho_R \cdot B_\rho$$

5. Write the following rescaling parameters to the product metadata

    a. For OLI:
        i. Reflectance Rescaling Gain, G_ρ
        ii. Reflectance Rescaling Bias, B_ρ
        iii. Radiance Rescaling Gain, G_L
        iv. Radiance Rescaling Bias, B_L

    b. For TIRS:
        i. Radiance Rescaling Gain, G_L
        ii. Radiance Rescaling Bias, B_L

7.4.22 Cloud Cover Assessment CCA – control

7.4.22.1 Background/Introduction

The Landsat Data Continuity Mission (LDCM) Cloud Cover Assessment (CCA) system will be tasked with creating full resolution cloud masks for every LDCM scene. Unlike previous Landsat cloud cover assessment algorithms, CCA for LDCM will be composed of several intermediate algorithms put together in a modular fashion.

7.4.22.2 Inputs

Descriptions	Units	Level	Source	Type
OLI scene data (L1G), as TOA reflectance	DN	Scene, all OLI bands		long
TIRS scene data (L1G), as TOA radiance	DN	Scene, all TIRS bands		long
Solar elevation angle	degrees	Scene	Metadata	float
Solar azimuth angle	degrees	Scene	Metadata	float
L1G rescaling LMAX	none (OLI) or radiance (TIRS)	N_{bands}	CPF or Metadata	float
L1G rescaling LMIN	none	N_{bands}	CPF or	float

	(OLI) or radiance (TIRS)		Metadata	
L1G rescaling QCALMAX	DN	N_{bands}	CPF or Metadata	long
L1G rescaling QCALMIN	DN	N_{bands}	CPF or Metadata	long
CCA Algorithm weights	none	$N_{algorithm}$ x $N_{classes}$	CPF or ODL file	float

Some intermediate CCA components may have additional inputs. These are discussed in the ADD for each CCA.

The CCA component algorithms may use any or all of the OLI and TIRS bands. The individual ADDs for these algorithms will refer to OLI Band numbers, which for clarity are named here:

OLI Band Number	Band Name	Equivalent Landsat 7 Band
1	Coastal Aerosol	n/a
2	Blue	1
3	Green	2
4	Red	3
5	NIR	4
6	SWIR 1	5
7	SWIR 2	7
8	Pan	8
9	Cirrus	n/a

TIRS bands are referred to as TIRS Band 1 and TIRS Band 2, and the Landsat 7 equivalent band for both is Band 6. (The equivalent Landsat 7 bands are provided as an aid to future CCA ADD writers and editors).

7.4.22.3 Outputs

The final output of the CCA system is a scene-wide cloud score that will be reported in the L1G metadata, and a quality band (QB) mask file – a 16 bit image of the same dimensions as the L1G scene.

Bit	Flag description	Values
0	Designated Fill	0 for image data 1 for fill data
1	Dropped Frame (Reserved)	0 for normal data 1 for dropped frame
2	Terrain Occlusion	0 for normal data 1 for terrain occlusion
3	Reserved	Reserved for a future 1-bit artifact designation.
4-5	Water confidence	00 = Not set. 01 = 0-35% confidence

		10 = 36-64% confidence 11 = 65-100% confidence
6-7	Reserved	Reserved for a future 2-bit class confidence designation.
8-9	Vegetation confidence (Reserved)	same as water confidence
10-11	Snow/Ice confidence	same as water confidence
12-13	Cirrus confidence	same as water confidence
14-15	Cloud confidence	same as water confidence

There are several reserved bits in the quality band mask file:

The 'Dropped Frame' flag depends on the existence of a recognizable fill pattern or reserved pixel value (or an artifact mask, which is not planned) for dropped frame data in the L1G image. If dropped frame data is resampled without a fill pattern or reserved pixel value, then bit 1 of the QB mask file will be left unused and will be reserved for a future 1-bit artifact designation.

Bit 3 is reserved for a possible future 1-bit artifact designation.

Bits 6 and 7 are reserved for a possible future 2-bit class confidence designation.

Bits 8 and 9 are a 2-bit confidence designation for pixels classified as Vegetation. The CCA algorithms that are known to be capable of creating a vegetation classification are AT-ACCA and ACCA, but they are poor classifiers for this class and their vegetation designations are not currently enabled. In the future it is expected that their classification will be refined and they will use these bits in the quality band mask. Also, there may be future CCA algorithms that perform vegetation classification. Until then, these bits are reserved.

The QB mask file format may change as development continues, but the basic structure will remain. Bits will be allocated for some artifacts that are distinguishable at the L1G stage of processing. Several classification types will exist and some range of confidence levels will be provided for each classification type.

The two-bit confidence levels are as follows:

00 No confidence level set. (Used for fill or for class not reported.)
01 Low confidence.
10 Mid confidence.
11 High confidence.

A QB value of 1 (00 01 hex) is reserved for fill data. It should not be possible to reach this QB value when processing a non-fill pixel.

In addition to the CCA final mask, each individual CCA component will create mask images, which may be held in memory or may be written to temporary files. A processing option should be available to save these mask image files for analysis. The format for the intermediate CCA mask image is the same as the final QB cloud mask structure, although the CCA components may leave many bits unused.

7.4.22.4 Options

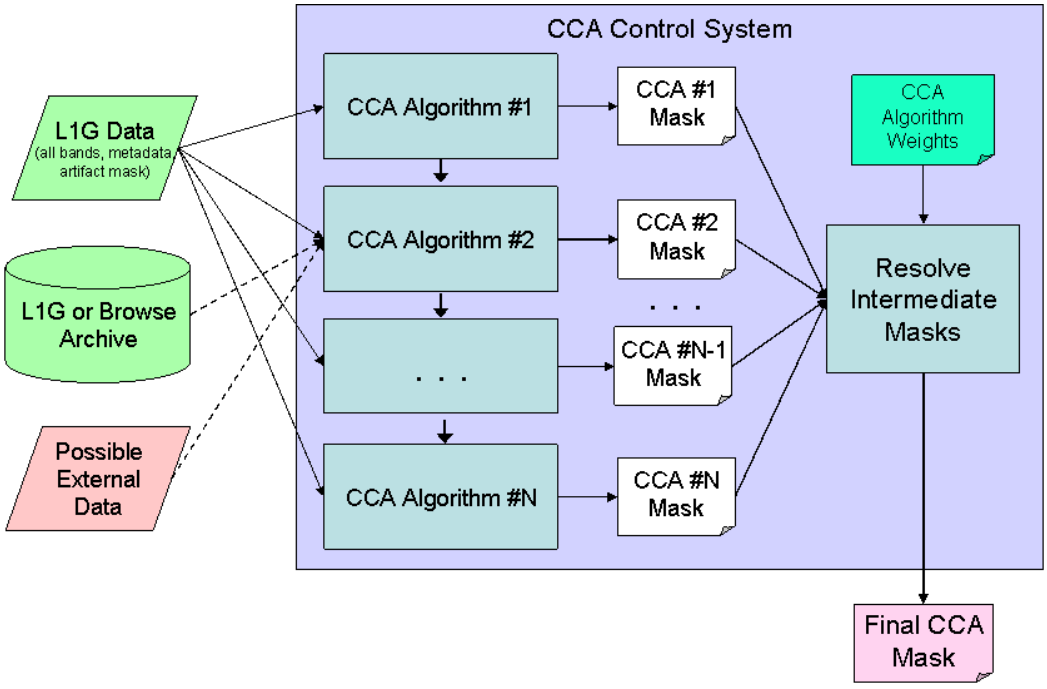
The CCA system will coordinate the intermediate CCA modules via the control parameters, which will exist in either the CPF or an ODL file. An example set of parameters is:

```
CCA_Algorithms = { "AT-ACCA", "See5-CCA", "ACCA", "Cirrus" }
CCA_Run_If_Thermal = { "No", "Always", "Yes", "Always" }

CCA_CLASS_TYPE = { "Cloud", "Cirrus", "Snow", "Veg", "Water" }
AT-ACCA_WEIGHTS = { 0.7, 0.0, 0.7, 0.0, 0.7 }
See5-CCA_WEIGHTS = { 1.0, 0.0, 0.0, 0.0, 0.0 }
ACCA_WEIGHTS = { 0.7, 0.0, 0.7, 0.0, 0.7 }
Cirrus_WEIGHTS = { 0.0, 1.0, 0.0, 0.0, 0.0 }

CREATE_INTERMEDIATE_CCA_MASKS = 0
CCA_INTERMEDIATE_MASK_FILE_NAMES = { "cca_atacca.img",
"cca_treecca.img", "cca_acca.img", "cca_cirrus.img" }
```

An example of CCA parameters.

7.4.22.5 Procedure

The CCA control system is a shell that runs, in any sequence, the intermediate CCA processes.



Flowchart describing the LDCM CCA control system.

First, the CCA algorithm weights are accessed to get a list of the CCA modules and their class-specific weights. Then each CCA module is called to create their intermediate cloud masks. Finally those intermediate masks are read back in and merged into a final cloud mask.

The detailed procedure is as follows:

1. The CCA algorithm weights are read in. This specifies the CCA algorithms to be run, the name of the intermediate cloud mask files, and the weights to be used later when merging them together.
2. The band image files and scene metadata are opened, and the solar angle information is read from the metadata.
3. The parameter 'CCA_Run_If_Thermal' specifies which CCA algorithms are dependent or contraindicated in the event that TIRS data is available.
    a. If the parameter is "No", do not run this CCA algorithm if TIRS data exists. If TIRS data do not exist, this algorithm should be run.
    b. If the parameter is "Yes", run this CCA algorithm only if TIRS data exists. If TIRS data do not exist, this algorithm should not be run.
    c. If the parameter is "Always", this CCA algorithm should always be run.
4. For each CCA algorithm to be run, an intermediate cloud mask is created:
    a. The intermediate mask output file is opened.
    b. For each pixel in the scene:
        i. One pixel value is read in from each band necessary for this CCA algorithm.
        ii. If any of the bands contain fill for this pixel, set the output mask value to fill (00 01 hex). Fill will be detected if a pixel value is equal to the reserved pixel value designating fill in L1G imagery.
        iii. If the pixel is not fill, the pixel value is converted to TOA reflectance (for OLI bands) or TOA radiance (for TIRS bands). Note that this must be the true TOA reflectance, including the scene-wide sun angle adjustment. (The LDCM L1G product is 'psuedo-reflectance', which can be converted to true TOA reflectance by dividing it by the sine of the solar elevation angle.)
        iv. The CCA evaluation function is called with the TOA values and the solar angle values for this pixel.
        v. Set this pixel in the output mask to the returned value from the evaluation function.
        vi. Write out the output mask pixel value.
    c. Once this CCA algorithm is finished processing, close this intermediate mask file and rewind all the band files used so the next CCA algorithm can start at the beginning of the scene.
5. When all CCA algorithms are finished processing, the band image files are closed. The intermediate CCA mask files are opened for input.
6. The resolver algorithm takes the pixel values from all intermediate CCA mask files and merges them into a final QB mask file:
    a. For each pixel in the scene:
        i. One pixel value is read in from each intermediate mask file.
        ii. If this pixel is fill (a reserved value of zero in any of the intermediate mask images), then set the final mask to a fill value and continue with (iv), below. No confidence scores need be calculated for this pixel.

iii. If this pixel is not fill, a score is created for this pixel for each class and confidence level. (This assumes that the weight is zero for classes not used by the associated algorithm).

1. For each class (water, snow, cloud, cirrus):

a. For each CCA algorithm's intermediate mask value:

i. If the CCA algorithm has set high confidence bits for this class, add this CCA algorithm's weight value to a high confidence score for this class.

ii. If the CCA algorithm has set mid confidence bits for this class, add this CCA algorithm's weight value to a mid confidence score for this class.

iii. If the CCA algorithm has set low confidence bits for this class, add this CCA algorithm's weight value to a low confidence score for this class.

b. The within-class confidence scores are then compared. The score with the largest value is designated as the confidence value in the final mask for this class. The mid confidence level wins any and all ties. (Thus, if the mid score is lowest, but the high and low scores are tied, the pixel is designated mid confidence.)

c. Set the appropriate bits in the final mask for the chosen class confidence value.

2. If this pixel has not been designated as fill, then a valid scene pixel count is incremented.

3. If this pixel has been designated as high confidence in either cloud or cirrus, then a cloud pixel count is incremented.

iv. If a L1G artifact mask is available and the artifact mask file indicates this pixel contains an artifact, set the artifact bits in the final mask. Currently a L1G artifact mask is not planned.

v. The final mask value for this pixel is written to the final QB mask file.

b. After all pixels have been processed, the scene-wide metadata cloud score for this scene is calculated by dividing the cloud pixel count by the valid scene pixel count.

7. When the resolver algorithm is finished, all open files are closed.

8. The intermediate CCA mask files are deleted.

7.4.23 Cloud Cover Assessment CCA – Artificial Thermal (AT)-ACCA

7.4.23.1 Background/Introduction

Cloud cover assessment for LDCM will be performed via a series of intermediate CCA algorithms, whose outputs will be resolved into a final mask. The Expanded AT-ACCA (Artificial Thermal Automated Cloud Cover Assessment) algorithm is one such intermediate process. It is a two-phase algorithm: The first phase is a decision tree based on the L7 ACCA algorithm, with the thermal band

replaced by a combination of reflective bands. The second phase is a voting schema to resolve ambiguous pixels. The output of these two algorithms are combined into an intermediate CCA mask.

7.4.23.2 Inputs

Descriptions	Units	Level	Source	Type
OLI Scene data (L1G), as TOA reflectance	none (reflectance)	Scene, OLI Bands 2-7.		float
Solar elevation angle	degrees	Scene	Metadata	float

7.4.23.3 Outputs

The output of each CCA component is an intermediate cloud mask file – a 16 bit image of the same dimensions as the L1Gs scene. The standardized format of the cloud mask file is:

Bit	Flag description	Values
0	Designated Fill	0 for image data 1 for fill data
1	Unused	
2	Unused	
3	Unused	
4-5	Water confidence	00 = Not set 01 = Low confidence 10 = Mid confidence 11 = High confidence
6-7	Unused	
8-9	Unused	
10-11	Snow/Ice confidence	same as water confidence
12-13	Unused	
14-15	Cloud confidence	same as water confidence

Cloud mask file bit format, as used by AT-ACCA.

A byte value of 1 (00 01 hex) is reserved for fill data. It should not be possible to reach this value when processing a non-fill pixel.

7.4.23.4 Procedure

The main loop of the CCA process opens the band files and output files, and reads information from the metadata. It then – for each non-fill pixel – passes the band reflectance values to the evaluation function for each CCA algorithm. The return value from the algorithm is written to the intermediate CCA mask file. The detailed procedure for the CCA main loop can be found in the CCA Control System ADD.

The Expanded AT-ACCA evaluation function is a two-phase algorithm based on the L7 ACCA algorithm.

The first phase of the AT-ACCA algorithm follows the structure of phase 1 of the Landsat 7 ACCA algorithm. Because the ACCA algorithm uses the L7 thermal band, in AT-ACCA an artificial thermal (AT) band is created from the OLI reflective bands. The AT band is then used with the other OLI bands to classify the scene pixels and assign a confidence score to them. This phase may classify a pixel as Water or Snow/Ice; flags for those classes are reserved in the cloud mask file structure. The AT-ACCA algorithm is also capable of Vegetation classification, but because the accuracy is poor it is not planned to use this algorithm for that purpose.

The second phase of Expanded AT-ACCA is known as the 'gD02' algorithm. It is a voting algorithm, intended to resolve some of the pixels designated as ambiguous by the earlier phases. The gD02 algorithm is a series of threshold tests using the reflective bands and the AT band. Each successful test carries one vote. 2 or more votes indicates that a pixel is clear, 0 votes indicates that it is cloudy, while 1 vote allows the ambiguous designation to stand.

The detailed procedure for the Expanded AT-ACCA evaluation function is:

1. Calculate the AT (Artificial Thermal) band.
   The AT band is calculated from the Landsat-like reflectance bands.

   AT = -92.7*ND(B4,B6) + 261.4*ND(B3,B7) - 48.8*ND(B3,B6) - 17.5*ND(B5,B3) - 146.9*ND(B2,B7) + 58.7*ND(B4,B2) - 117*ND(B3,B2) + 172*CSA*B6 + 76*CSA*B5 + 151*CSA*B4 - 951*CSA*B3 + 539*CSA*B2 + 28*B7 - 132*B6 - 106.2*B5 - 22.4*B4 + 633.1*B3 - 443.6*B2 + 302.0986

   where ND(x,y) = The normalized difference between x and y.
   = (x – y)/(x + y)
   CSA = The cosine of the solar zenith angle.
   (Note that the metadata may only provide the solar elevation angle. Zenith = 90 degrees – Elevation.)
   Bx = The reflectance in OLI band x.
   AT = The calculated AT band value.

   The equation used to calculate the AT band was arrived at by an empirical fit to a test set of over 1 million pixels of L7 data. It has no scientific derivation.
   Prototype code for the AT band calculation is provided in the cubist_therm.c routine.
2. Phase 1: Perform the AT-ACCA cloud detection algorithm.
   a. B4 test #1: If B4 > 0.08, go to 2.a.i.
      i. NDSI test #1: If ND(B3,B6) > -0.25 and < 0.7, go to step 2.a.i.1. (ND(B3,B6) is known as NDSI, the Normalized Differential Snow Index.)
         1. AT test: If AT < 300, go to 2.a.i.1.a.
            a. B6composite test: If (1-B6)*AT < 225, go to step 2.a.i.1.a.i.
               i. B54 ratio test: If B5/B4 < 2.25, go to step 2.a.i.1.a.i.1.
                  1. B53 ratio test: If B5/B3 < 2.2, go to step 2.a.i.1.a.i.1.a.
                     a. B56 ratio test: If B5/B6 > 1, then this pixel is a cloud. Set the output value to high confidence cloud. (CCA_INT_CLOUD_HIGH)

ii. If any of the B56, B53, or B54 ratio tests fail, the pixel is temporarily designated as ambiguous, and proceed to Phase 2.

b. If the B6composite test fails, perform the B6 test: If B6 < 0.08, then this pixel is clear. Set the output value to clear of unknown class. (Low value for cloud confidence and low value for class confidence. This is set by the value CCA_INT_UNKNOWN_LOW + CCA_INT_CLOUD_LOW.)

c. If the B6 test fails, temporarily designate the output value as ambiguous, and proceed to Phase 2.

2. If the AT test fails, this pixel is clear. Set the output value to clear. (CCA_INT_UNKNOWN_LOW + CCA_INT_CLOUD_LOW)

ii. If the NDSI test #1 fails, perform NDSI test #2: If ND(B3,B6) > 0.8, then the pixel is snow or ice. Set the output value to high confidence snow. (CCA_INT_SNOW_HI + CCA_INT_CLOUD_LOW)

iii. If NDSI test #2 fails, the pixel is not snow but is not a cloud. Set the output value to clear. (CCA_INT_UNKNOWN_LOW + CCA_INT_CLOUD_LOW)

b. If B4 test #1 fails, perform B4 test #2: If B4 < 0.07, then the pixel is water. Set the output value to mid confidence water. (CCA_INT_WATER_MID + CCA_INT_CLOUD_LOW) The confidence is set to mid instead of high because the ACCA algorithm does not do a good job at water discrimination, so this test is not very effective.

c. If B4 test #2 fails, the pixel is not water but is not a cloud. Temporarily designate the output value as ambiguous, and proceed to Phase 2.

3. If the output from Phase 1 is non-ambiguous, skip Phase 2.

4. Phase 2: gD02 tests. The gD02 tests are a series of threshold tests arrived at by statistical analysis of L7 data. They have no scientific derivation. They are only run when the output of the AT-ACCA algorithm (Phase 2) is ambiguous.

a. Before running the gD02 tests, A 'g score' is initialized to zero. Every successful test increments the g score by 1.

b. The gD02 tests can be run in any sequence. All of them must be performed. The tests are:

	Test is successful if parameter	
Test Parameter	**is less than**	**or is greater than**
B2	0.140	*n/a*
B3	0.111	*n/a*
B4	0.093	*n/a*
B6/nfac	0.087	0.481
B4/B2	0.640	1.034
ND(CSA*B2,B5)	-0.454	0.262
ND(B2,B6)	-0.138	0.716
CSA*B2/B7	0.736	3.914
B4/B3	0.810	1.075
ND(B3,B5)	-0.404	0.160
ND(B3,B6)	-0.186	0.716
ND(B3,B7)	-0.018	0.754

ND(CSA*B4,B5)	-0.566	-0.016
ND(B4,B6)	-0.232	0.692
ND(B4,B7)	-0.030	0.738
ND(B6,B7)	-0.050	0.300

where nfac = The normalization factor for OLI bands 2 through 7. This is the square root of the sum of the square of the band reflectances.

$$= \sqrt{\sum_{x=2}^{x=7} Bx^2}$$

c. When all tests have been performed, the g score is examined. If it is equal to 0 (none of the tests were successful) then the pixel is a high confidence cloud (CCA_INT_CLOUD_HIGH). If the g score is 2 or larger (2 or more of the tests were successful), the pixel is clear (CCA_INT_UNKNOWN_LOW + CCA_INT_CLOUD_LOW). For all other outcomes (g=1), the pixel remains ambiguous.
d. If the pixel is still considered to be ambiguous, it is set to mid confidence cloud. (CCA_INT_CLOUD_MID)

5. Return the output value.

7.4.23.5 Cloud Cover Assessment - See5

7.4.23.6 Background/Introduction

Cloud cover assessment for LDCM will be performed via a series of intermediate CCA algorithms, whose outputs will be resolved into a final mask. The See5 CCA algorithm is one such intermediate process. It is a decision tree test using thresholds derived with the COTS package C5 by Rulequest Research. The C5 software is a development tool only – it is not required to run the See5 CCA module, nor would it be used in operational verification of the algorithms.

7.4.23.7 Inputs

Descriptions	Units	Level	Source	Type
OLI Scene data (L1G), as TOA reflectance	none (reflectance)	Scene, OLI Bands 2-7.		float
Solar elevation angle	degrees	Scene	Metadata	float

7.4.23.8 Outputs

The output of each CCA component is an intermediate cloud mask file – a 16 bit image of the same dimensions as the L1Gs scene. The standardized format of the cloud mask file is:

Bit	Flag description	Values
0	Designated Fill	0 for image data 1 for fill data
1	Unused	

2	Unused	
3	Unused	
4-5	Unused	
6-7	Unused	
8-9	Unused	
10-11	Unused	
12-13	Cirrus confidence	00 = Not set 01 = Low confidence 10 = Mid confidence 11 = High confidence
14-15	Cloud confidence	same as cirrus confidence

Cloud mask file bit format, as used by See5 CCA.

A byte value of 1 (00 01 hex) is reserved for fill data. It should not be possible to reach this value when processing a non-fill pixel.

7.4.23.9 Procedure

The main loop of the CCA process opens the band files and output files, and reads information from the metadata. It then – for each non-fill pixel – passes the band reflectance values to the evaluation function for each CCA algorithm. The return value from the algorithm is written to the intermediate CCA mask file. The detailed procedure for the CCA main loop can be found in the CCA Control System ADD.

The See5 evaluation function is a complicated decision tree with 245 branches. It was created using the COTS package C5 by Rulequest Research. The package takes input data – with no a priori knowledge of the data's contents or physical meaning – and splits the data into sections that minimize the information entropy in the dataset. It then splits those sections into smaller and smaller subsections, creating a decision tree that can be used to classify the dataset into a user-defined number of branches.

The See5 CCA function was created with a 6.5 million point dataset derived from a global selection of Landsat 7 scenes, and was constrained to less than 255 branches in the final decision tree for debugging purposes. The function uses as input OLI bands 2-7 and the solar elevation angle from the metadata. Validation on a 4 billion point global dataset has shown that the See5 CCA function is 89% accurate in the detection of clouds.

Because it was derived from a simple statistical model with no knowledge of radiometry, the See5 CCA algorithm has no scientific derivation and is not meant to be comprehended as anything but a large decision tree. The function presented may be replaced with a new one in the future, when the characteristics of the OLI instrument become known and the model can be refined.

7.4.24 Automated Cloud Cover Assessment ACCA

7.4.24.1 Background/Introduction

Cloud cover assessment for LDCM will be performed via a series of intermediate CCA algorithms, whose outputs will be resolved into a final mask. The ACCA (Automated Cloud Cover Assessment) algorithm is one such intermediate process. It is a decision tree based on the L7 ACCA algorithm. The output of the ACCA algorithms is an intermediate CCA mask.

7.4.24.2 Inputs

Descriptions	Units	Level	Source	Type
OLI Scene data (L1G), as TOA reflectance	none (reflectance)	Scene, OLI Bands 2-7.		float
TIRS scene data (L1G), as TOA radiance	radiance	Scene, TIRS Band 1.		float

7.4.24.3 Outputs

The output of each CCA component is an intermediate cloud mask file – a 16 bit image of the same dimensions as the L1Gs scene. The standardized format of the cloud mask file is:

Bit	Flag description	Values
0	Designated Fill	0 for image data 1 for fill data
1	Unused	
2	Unused	
3	Unused	
4-5	Water confidence	00 = Not set 01 = Low confidence 10 = Mid confidence 11 = High confidence
6-7	Unused	
8-9	Unused	
10-11	Snow/Ice confidence	same as water confidence
12-13	Unused	
14-15	Cloud confidence	same as water confidence

Cloud mask file bit format, as used by ACCA.

A byte value of 1 (00 01 hex) is reserved for fill data. It should not be possible to reach this value when processing a non-fill pixel.

7.4.24.4 Procedure

The main loop of the CCA process opens the band files and output files, and reads information from the metadata. It then – for each non-fill pixel – passes the band reflectance values to the evaluation function for each CCA algorithm. The return value from the algorithm is written to the intermediate CCA mask file. The detailed procedure for the CCA main loop can be found in the CCA Control System ADD.

The ACCA algorithm follows the structure of phase 1 of the Landsat 7 ACCA algorithm. In addition to clouds, this algorithm may classify a pixel as Water or Snow; flags for those classes are reserved in the intermediate cloud mask file structure. The ACCA algorithm is also capable of Vegetation classification, but because the accuracy is poor it is not planned to use this algorithm for that purpose.

The ACCA algorithm makes use of the following band-derived values:

Bx	=	The reflectance in OLI band x.
T	=	The radiance of the thermal band.
ND(x,y)	=	The normalized difference between x and y.
	=	$(x - y)/(x + y)$

The procedure for the ACCA evaluation function is:

6. Phase 1: Perform the ACCA cloud detection algorithm.
    a. B4 test #1: If B4 > 0.08, go to step 1.a.i.
        i. NDSI test #1: If ND(B3,B6) > -0.25 and < 0.7, go to step 1.a.i.1. (ND(B3,B6) is known as NDSI, the Normalized Differential Snow Index.)
            1. Thermal test: If T < 9.390745, go to step 1.a.i.1.a.
                a. B6composite test: If (T < 666.09/(exp(5.70093*(1-B5) - 1)), go to step 1.a.i.1.a.i.
                    i. B54 ratio test: If B5/B4 < 2.25, go to step 1.a.i.1.a.i.1.
                        1. B53 ratio test: If B5/B3 < 2.2, go to step 1.a.i.1.a.i.1.a.
                            a. B56 ratio test: If B5/B6 > 1, then this pixel is a cloud. Set the output value to high confidence cloud. (CCA_INT_CLOUD_HIGH)
                    ii. If any of the B56, B53, or B54 ratio tests fail, the pixel is designated as ambiguous. (CCA_INT_CLOUD_MID)
                b. If the B6composite test fails, perform the B6 test: If B6 < 0.08, then this pixel is clear. Set the output value to clear of unknown class. (Low value for cloud confidence and low value for class confidence. This is set by the value CCA_INT_UNKNOWN_LOW + CCA_INT_CLOUD_LOW.)
                c. If the B6 test fails, designate the output value as ambiguous. (CCA_INT_CLOUD_MID)
            2. If the thermal test fails, this pixel is clear. Set the output value to clear. (CCA_INT_UNKNOWN_LOW + CCA_INT_CLOUD_LOW)
        ii. If the NDSI test #1 fails, perform NDSI test #2: If ND(B3,B6) > 0.8, then the pixel is snow or ice. Set the output value to high confidence snow. (CCA_INT_SNOW_HI + CCA_INT_CLOUD_LOW)
        iii. If NDSI test #2 fails, the pixel is not snow but is not a cloud. Set the output value to clear. (CCA_INT_UNKNOWN_LOW + CCA_INT_CLOUD_LOW)

   b. If B4 test #1 fails, perform B4 test #2: If B4 < 0.07, then the pixel is water. Set the output value to mid confidence water. (CCA_INT_WATER_MID + CCA_INT_CLOUD_LOW) The confidence is set to mid instead of high because the ACCA algorithm does not do a good job at water discrimination, so this test is not very effective.
   c. If B4 test #2 fails, the pixel is not water but is not a cloud. Designate the output value as ambiguous. (CCA_INT_CLOUD_MID)
7. Return the output value.

7.4.24.5 Maturity

Level 1. Changes will be made to this procedure when the characteristics of the OLI and TIRS instruments are known.

Other possible changes that may occur in the ACCA algorithm are:

- Disambiguation – A disambiguation algorithm may be appended to the output of the ACCA algorithm, to clean up the ambiguous results. This disambiguation algorithm may be the gD02 algorithm used in Expanded AT-ACCA, or may be a new algorithm designed for use with LDCM ACCA.
- Surface Temperature – The ACCA algorithm may be re-engineered to use a split window algorithm, which uses two thermal bands to calculate an approximate surface temperature. This will require converting two of the TIRS bands into a surface temperature before passing through the ACCA decision tree. The surface temperature algorithm is TBD.
- Whole Scene Processing – The above procedure assumes that each pixel is read in, processed, and written back out in sequence. This method uses less system memory and is parallelizable. However, for performance reasons, it may be desirable to read in all (or a portion) of the bands and process them in memory before writing out the mask. This will require changes to the main loop process flow and to the process flow of the CCA control system.
- Classification Improvements – The AT-ACCA algorithm can be altered to output Vegetation classification for each pixel in the cloud mask. This is not expected to be studied in detail until after the launch of LDCM.
- Band 2 – This ADD has been specified with TIRS Band 1 as the input thermal data. As of this writing it is not clear which band will be best for deriving thermal brightness temperature. In the future TIRS Band 1 may be replaced with TIRS Band 2 to improve the algorithm.
- Additional parameters – Any of the above changes to this algorithm may require additional input parameters, most likely including (but not limited to) the Solar Elevation Angle.

7.4.25 Cloud Cover Assessment CCA- See5

7.4.25.1 Background/Introduction

Cloud cover assessment for LDCM will be performed via a series of intermediate CCA algorithms, whose outputs will be resolved into a final mask. The See5 CCA algorithm is one such intermediate process. It is a decision tree test using thresholds derived with the COTS package C5 by Rulequest Research. The C5 software is a development tool only – it is not required to run the See5 CCA module, nor would it be used in operational verification of the algorithms.

7.4.25.2 Inputs

Descriptions	Units	Level	Source	Type
OLI Scene data (L1G), as TOA reflectance	none (reflectance)	Scene, OLI Bands 2-7.		float
Solar elevation angle	degrees	Scene	Metadata	float

7.4.25.3 Outputs

The output of each CCA component is an intermediate cloud mask file – a 16 bit image of the same dimensions as the L1Gs scene. The standardized format of the cloud mask file is:

Bit	Flag description	Values
0	Designated Fill	0 for image data 1 for fill data
1	Unused	
2	Unused	
3	Unused	
4-5	Unused	
6-7	Unused	
8-9	Unused	
10-11	Unused	
12-13	Cirrus confidence	00 = Not set 01 = Low confidence 10 = Mid confidence 11 = High confidence
14-15	Cloud confidence	same as cirrus confidence

Cloud mask file bit format, as used by See5 CCA.

A byte value of 1 (00 01 hex) is reserved for fill data. It should not be possible to reach this value when processing a non-fill pixel.

7.4.25.4 Procedure

The main loop of the CCA process opens the band files and output files, and reads information from the metadata. It then – for each non-fill pixel – passes the band reflectance values to the evaluation

function for each CCA algorithm. The return value from the algorithm is written to the intermediate CCA mask file. The detailed procedure for the CCA main loop can be found in the CCA Control System ADD.

The See5 evaluation function is a complicated decision tree with 245 branches. It was created using the COTS package C5 by Rulequest Research. The package takes input data – with no a priori knowledge of the data's contents or physical meaning – and splits the data into sections that minimize the information entropy in the dataset. It then splits those sections into smaller and smaller subsections, creating a decision tree that can be used to classify the dataset into a user-defined number of branches.

The See5 CCA function was created with a 6.5 million point dataset derived from a global selection of Landsat 7 scenes, and was constrained to less than 255 branches in the final decision tree for debugging purposes. The function uses as input OLI bands 2-7 and the solar elevation angle from the metadata. Validation on a 4 billion point global dataset has shown that the See5 CCA function is 89% accurate in the detection of clouds.

Because it was derived from a simple statistical model with no knowledge of radiometry, the See5 CCA algorithm has no scientific derivation and is not meant to be comprehended as anything but a large decision tree. The function presented may be replaced with a new one in the future, when the characteristics of the OLI instrument become known and the model can be refined.

7.4.25.5 Verification Methods

The See5 CCA prototype code will be used to generate cloud masks for a standardized set of data. Masks created by the operational algorithm will be verified by comparison with the prototype masks of the same data set, and by manual inspection with the imagery to verify its accuracy in cloud detection.

7.4.25.6 Maturity

Level 2. The format and operation of the See5 CCA algorithm will not change. The algorithm functions themselves may change when the characteristics of the OLI instrument are known.

Other possible changes that may occur in the See5 CCA algorithm are:

- Debug Output – Because of the complexity of the See5 decision trees, it would be useful to have an additional byte image created for debug purposes. The pixel values in this debug image would correspond to the index of the decision tree branches that evaluated the pixel. This would require changes to the main loop (to open a new file, and to pass data to and from the evaluation functions as a structure, so both cloud mask and debug data is returned) and to the evaluation functions. Prototype code with debug output enabled is available, but is not presented in this ADD. Because code already exists, the main impact of this option lies in writing out a new debug image; the effort required to modify the code would be negligible.
- Saturation Splitting – The evaluation functions given in this ADD were created over exclusively non-saturated L7 data. It is expected that OLI will not have widespread saturation in bands 2-7. (Small areas of saturation due to fires or specular solar reflection

will not adversely affect cloud scores.) If saturation turns out to be a major issue with OLI data, then the See5 CCA algorithm will need to be broken into two functions – one for saturated data and one for non-saturated data – and a switch will need to be implemented to call the correct function based on the amount of saturation in the data. This will require adding a saturated data algorithm, and a wrapper for both algorithms that will detect saturation and call the appropriate function.

- Whole Scene Processing – The above procedure assumes that each pixel is read in, processed, and written back out in sequence. This method uses less system memory and is parallelizable. However, for performance reasons, it may be desirable to read in all (or a portion) of the bands and process them in memory before writing out the mask. This will require changes to the main loop process flow and to the process flow of the CCA control system, which must be retained, and an option provided in the CCA Control file to flag the correct form of processing for each CCA algorithm.
- Classification Improvements – The See5 CCA algorithm can be altered to output a predicted classification for each pixel in the cloud mask. Currently, other classifications in the cloud mask format include Water, Vegetation, and Snow/Ice. The current implementation of See5 does not use them, but it could if the requisite training data becomes available. In addition, Cirrus classification in the See5 CCA algorithm will likely be improved once OLI data becomes available. Detailed classification is not expected to be implemented by the See5 CCA algorithm by the launch of LDCM.
- Land Cover Lookup – The See5 CCA algorithm can take a global land cover database as additional input, with per-pixel latitude/longitude coordinate labels, thus giving the algorithm more information by which to classify pixels. This would require a global database to be linked to the Level 1 processing system. It is not expected to be implemented by the launch of LDCM.

7.4.26 Cloud Cover Assessment CCA – Cirrus

7.4.26.1 Background/Introduction

Cloud cover assessment for LDCM will be performed via a series of intermediate CCA algorithms, whose outputs will be resolved into a final mask. The Cirrus algorithm is one such intermediate process that tests for Cirrus clouds using the OLI cirrus band.

7.4.26.2 Inputs

Descriptions	Units	Level	Source	Type
OLI Band 9 (Cirrus) Scene data (L1G), as TOA reflectance	none (reflectance)	Scene, OLI band 9 (Cirrus) only.		float
Cirrus Threshold	none		CPF	float

The Cirrus Threshold is a CPF parameter that has a default value of 0.02. It may change after launch.

7.4.26.3 Outputs

The output of each CCA component is an intermediate cloud mask file – a 16 bit image of the same dimensions as the L1G scene. The standardized format of the cloud mask file is:

Bit	Flag description	Values
0	Designated Fill	0 for image data 1 for fill data
1	Unused	
2	Unused	
3	Unused	
4-5	Unused	
6-7	Unused	
8-9	Unused	
10-11	Unused	
12-13	Cirrus confidence	00 = Not set. 01 = Low confidence 10 = Mid confidence (currently unused) 11 = High confidence
14-15	Unused	

Cloud mask file bit format, as used by Cirrus CCA.

A byte value of 1 (00 01 hex) is reserved for fill data. It should not be possible to reach this value when processing a non-fill pixel.
The 'Mid' confidence level is currently unused by the Cirrus algorithm, but it is retained in the cloud mask file format for possible future use.

7.4.26.4 Procedure

The main loop of the CCA process opens the band files and output files, and reads information from the metadata. It then – for each non-fill pixel – passes the band reflectance values to the evaluation function for each CCA algorithm. The return value from the algorithm is written to the intermediate CCA mask file. The detailed procedure for the CCA main loop can be found in the CCA Control System ADD.

The Cirrus evaluation function is a simple threshold test, with a threshold value defined by a CPF parameter. The procedure for the Cirrus evaluation function is:

8. Perform the Cirrus detection algorithm.
    a. If OLI Band 9 (Cirrus band) reflectance > the Cirrus threshold from the CPF, then add the high confidence cirrus flag to the output value. (CCA_INT_CIRRUS_HIGH)
    b. Otherwise, add the low confidence cirrus flag to the output value. (CCA_INT_CIRRUS_LOW)
9. Return the output value.

7.4.26.5 Maturity

Level 1. Changes will be made to this procedure when the characteristics of the OLI instrument are known.

Other possible changes that may occur in the Cirrus algorithm are:

- Algorithm Refinement – The cirrus evaluation function may change once the characteristics of the OLI instrument are known.
- Whole Scene Processing – The above PDL for the main loop assumes that each pixel is read in, processed, and written back out in sequence. This method uses less system memory and is parallelizable. However, for performance reasons, it may be desirable to read in all (or a portion) of the bands and process them in memory before writing out the mask. This will require changes to the main loop process flow and to the process flow of the CCA control system..

7.5 OLI Radiometry Algorithms

7.5.1 OLI Bias Model Calibration

7.5.1.1 Background

The focal plane assembly for the OLI includes 14 sensor chip assemblies (SCA's) (or focal plane modules (FPM's) when the bandpass filters are included). Each SCA contains the active detector elements for all the 9 imaging bands and a blind band, plus video reference pixels (VRP's) at both ends of each row of active detectors. The VRP's consist of all the electronics of a normal active detector minus the detector itself, which is replaced by a capacitor. These VRP's thus are sensitive to most of the same electronic variation as the active pixels, minus photo sensitivity. There are 6 VRP's at each end of each row for a total of 12 per band, with the exception of the Pan band which has 12 at each end for a total of 24. Although included as a design feature, these VRP's were not indicated for use in the initial algorithm description for the bias model. The plan had been to use the blind band (full HgCdTe detectors, though masked from light), to track the dark response of the HgCdTe detectors. Long dark collects taken during Engineering Design Unit (EDU) as well as Flight Unit testing did not show a good correlation of these blind detectors' response to the active response in the absence of light. However, frame to frame variation as well as variation in the collect to collect mean in the dark response of the active pixels was observed to be tracked well by the VRPs. Since these data indicate that the frame to frame and scene to scene behavior of the VRPs is related to that of the imaging detectors, the data from them are used to estimate a per frame bias for the imaging detectors.

This algorithm calculates three sets of coefficients to estimate the bias of the VNIR and SWIR imaging detectors. First, a per-band, per-SCA scaling factor is calculated that relates the frame to frame behavior of the VRP data to that of the dark response of the imaging detectors. The coefficient is the slope resulting from a linear regression of the per-band, per-SCA, per-frame averages of the imaging detector data and the per-band, per-SCA, per-frame averages of the VRP data from N_0 shutter collects (excluding long shutter collects) acquired near (in time) to the Earth acquisition being

corrected. This coefficient is used to estimate the frame to frame variation of the bias within an image. Second, two coefficients, a scaling factor and a constant are determined which are used to estimate the mean of the dark response within an image. These coefficients are from a linear regression of the per FPM means of VRP data and the per-detector means of the imaging detector data from of N_1 shutter collects (excluding long shutter collects) acquired near (in time) to the Earth acquisition being corrected. Saturated pixels and impulse noise pixels, as well as pixels from dropped frames and inoperable detectors, are rejected from the data (both imaging and VRP) prior to fitting the selected model. This algorithm is anticipated to be run for every Earth acquisition as part of processing to produce a level 1R product. Note that the shutter collects included in the N_0 collects are also included in what is anticipated to be the larger group of N_1 shutter collects, and again also that no long shutter collects will be used in this algorithm.

This algorithm should also be implemented as a stand-alone process to enable bias model parameters to be determined for an arbitrary date/time range. This allows for testing the accuracy of the bias estimates.

7.5.1.2 Input

Descriptions	Symbol	Unit	Level	Source	Type
Impulse Noise locations in VRPs in shutter collects†			N_0 x N_{bands} x N_{SCAs} x N_{VRPs} x N_{lines}	Mask	Integer
VRP Operability List†			N_1 x N_{bands} x N_{SCAs} x N_{VRPs}	CPF	Integer
Saturated VRP locations in shutter collects†			N_0 x N_{bands} x N_{SCAs} x N_{VRPs} x N_{lines}	Mask	Integer
Dropped Line locations in VRPs in shutter collects†			N_0 x N_{bands} x N_{SCAs} x N_{VRP} x N_{lines}	Mask	Integer
Impulse Noise locations in shutter collects†			N_0 x N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{lines}	Mask	Integer
Saturated Pixel locations in shutter collects†			N_0 x N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{lines}	Mask	Integer
Dropped Line locations in shutter collects†			N_0 x N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{lines}	Mask	Integer
Shutter Histogram Statistics for N_1 collects immediately preceding current interval and one collect immediately after VRP data included	S	DN	(N_1+1)x$(N_{bands}+1)$x N_{SCAs} x $N_{detectors}$	Histogram Statistics	Float
Cross track averages of VRP data from shutter collects	A_{VRP}	DN	N_1 x$(N_{bands}+1)$x N_{SCAs} x N_{lines}	Histogram Statistics	Float
Shutter Data-VRP included	Q and Q_{VRP}	DN	N_0 x N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{lines}	L0R	Integer

Nominal Integration Times (MS and Pan)		Micro-seconds	2	CPF	Integer
Collection Integration Times (MS and Pan)		Milli-seconds	2	L0R	Float

*for N pre-acquisition shutter acquisitions; includes VRP data as well. N_{bands}+1 due to the Pan band.
N_{bands} does not include the blind band

7.5.1.3 Output

Descriptions	Symbol	Unit	Level	Target	Type
Pre-acquisition shutter average (S_A)*	S_A*	DN	(N_{bands} +1) x N_{SCAs} x $N_{detectors}$	BPF	Float
Post-acquisition shutter average (S_B)*	S_B*	DN	(N_{bands} +1) x N_{SCAs} x $N_{detectors}$	BPF	Float
Per-SCA cross-track VRP averages	A_{VRP}	DN	(N_{bands} +1) x N_{SCAs} x N_{frames}	DB	Float
Bias Model Parameter (a_0)*	a_0	N/A	(N_{bands}+1) x N_{SCAs}	BPF	Float
Bias Model Parameter (a_1)*	a_1	N/A	(N_{bands} +1) x N_{SCAs} x $N_{detectors}$	BPF	Float
Bias Model Parameter (C_1)*	C_1	N/A	(N_{bands} +1) x N_{SCAs} x $N_{detectors}$	BPF	Float
R-squared values from calculating a_0 (R_0^2)*	R_0^2	N/A	(N_{bands} +1) x N_{SCAs}	Characterization DB	Float
R-squared values from calculating a_1 and C_1 (R_1^2)*	R_1^2	N/A	(N_{bands} +1) x N_{SCAs}	Characterization DB	Float

*N_{bands}+1 accounts for the separation of odd and even lines in the Pan band
N_{bands} does not include the blind band

7.5.1.4 Options

- Number of pre-acquisition shutter collects (N_0) and shutter averages (N_1) used for bias model parameter determination (default is N_0=1 and N_1=40)
- Output BPF to file (default off)
- Start and stop date/time of desired bias model parameters (T0 and T1)
    - Normally this is the date/time of the scene being processed.
- The standalone version of this code should have the following options. Two, and only 2 of the 3 options must be chosen.
    - Start date for scenes to be included in N_1
    - End date for scenes to be included in N_1
    - Number of scenes to be included in N_1

  The newest scene within the selected range of scenes will be used for N_0.

7.5.1.5 Procedure

This algorithm can be divided into three sections. In the first section statistics needed for calculations done later in the algorithm are calculated or pulled from histogram statistics and stored to the database. In the second section the coefficient a_0 which is used for per frame correction is calculated. In the third section the coefficients used to estimate the per detector bias mean, a_1 and C_1 are calculated.

Section 1.

For all bands

1. Compare the integration time of the data with nominal integration time. Proceed if they are the same.
2. Retrieve the pre-and post-acquisition shutter histogram statistics collected prior to the desired start date/time (T0) and after the desired stop date/time (T1). These statistics are included in *S*, but are now known as S_A and S_B.
3. Send S_A and S_B to the BPF database. This is all that is done with S_B in this algorithm, although S_A is used in later steps.
4. Retrieve N_0 shutter collects (*Q* and Q_{VRP}) prior to the desired start date/time (T0). The first shutter collect before the desired start time will be the pre-acquisition shutter collect. Note that only nominal integration times are used as shutter collects taken during integration time sweeps will vary. Also note that anomalous pixels are excluded from ALL calculations in this algorithm. Pixels from inoperable detectors and VRPs are considered anomalous pixels.
5. For each band (except for the pan band) and for each SCA, and treating the data from all selected collects as a single data set
   a. Find the cross-track average of all of the VRP detectors.

   $$A_{VRP}(b,s,f)=\frac{1}{N_d}\sum_{d=1}^{N_d}Q_{VRP}(b,s,d,f) \quad (1)$$

   where *f* is line, Q_{VRP} is the VRP detector response for the specific band, N_d is the number of operable VRP detectors in a band, and A_{VRP} is the VRP average for frame *f*.

   b. Find the cross-track average of all of the imaging detectors.

   $$A(b,s,f)=\frac{1}{N_d}\sum_{d=1}^{N_d}Q(b,s,d,f) \quad (2)$$

   where *Q* is the imaging detector response for a specific band, N_d here is the number of operable imaging detectors in the specific band, and *A* is the imaging detector average for frame *f.*.
6. For the pan band, calculate cross track averages in the same way as in (1) and (2), only treat odd and even frames separately.

$$A_{VRP,even}(b,s,f)=\frac{1}{N_d}\sum_{d=1}^{N_d}Q_{VRP}(b,s,d,f),\ f=2,4,6,... \quad (3)$$

$$A_{even}(b,s,f)=\frac{1}{N_d}\sum_{d=1}^{N_d}Q(b,s,d,f),\ f=2,4,6,... \quad (4)$$

$$A_{VRP,odd}(b,s,f)=\frac{1}{N_d}\sum_{d=1}^{N_d}Q_{VRP}(b,s,d,f),\ f=1,3,5,... \quad (5)$$

$$A_{odd}(b,s,f)=\frac{1}{N_d}\sum_{d=1}^{N_d}Q(b,s,d,f),\ f=1,3,5,... \quad (6)$$

7. Write the per-SCA VRP cross-track averages to the database.
8. Retrieve N_1 collects of per detector means (S) and per frame VRP averages (A_{VRP}) from histogram statistics from before the desired start date/time. This includes S_A. Note that all N_1 collects should have been collected at nominal integration time.
9. For each band (except for the Pan band) and each of the N_1 collects, find the per-SCA mean of the VRP data.

$$\overline{A_{VRP}}(b,s,c)=\frac{1}{N_f}\sum_{f=1}^{N_f}A_{VRP}(c,b,s,f) \quad (7)$$

where A_{VRP} here is the per detector VRP average from histogram statistics, N_f is the number of frames, and c is collect.

10. Calculate the per-SCA VRP means for the pan band in the same way as (7), only treat the averages from odd and even frames separately.

$$\overline{A_{VRP,even}}(b,s,c)=\frac{1}{N_f}\sum_{f=1}^{N_f}A_{VRP,even}(c,b,s,f),\ f=2,4,6,... \quad (8)$$

$$\overline{A_{VRP,odd}}(b,s,c)=\frac{1}{N_f}\sum_{f=1}^{N_f}A_{VRP,odd}(c,b,s,f),\ f=1,3,5,... \quad (9)$$

Section 2.

1. For each band (except for the pan band) and for each SCA, and treating the data from all selected collects as a single data set
   a. Using a least squares fit, find the coefficient a_0 that relates A_{VRP} to A.
   $$A(b,s,f)\approx a_0(b,s)A_{VRP}(b,s,f) \quad (10)$$
   An example of this is plotted in Figure 1. The offset determined by the linear regression is ignored, except in calculating the R-squared values.

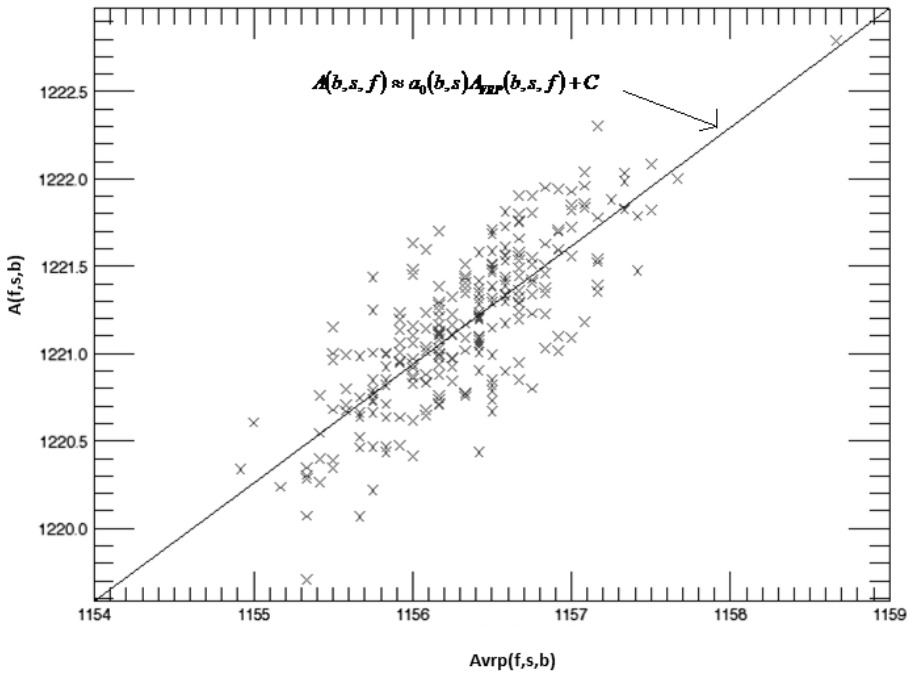


Figure 1. VRP cross track average plotted against the imaging detector cross track average for one SCA of one band

b. Calculate the R-squared values of the fit, R_0^2 $A_{estimate}(b,s,f) = a_0(b,s)A_{VRP}(b,s,f) + C$ (11)

$$SS_{err}(b,s) = \sum_{f=1}^{N_f} (A_{estimate}(b,s,f) - A(b,s,f))^2 \quad (12)$$

$$\overline{A}(b,s) = \frac{1}{N_f}\sum_{f=1}^{N_f} A(b,s,f) \quad (13)$$

$$SS_{tot}(b,s) = \sum_{f=1}^{N_f} \left(A(b,s,f) - \overline{A}(b,s)\right)^2 \quad (14)$$

$$R_0{}^2(b,s) = 1 - \frac{SS_{err}(b,s)}{SS_{tot}(b,s)} \quad (15)$$

where $A_{estimate}$ is the estimate of the of the A, N_f is the number of frames, and C is the offset calculated during the linear regression. The value will no longer be used after these calculations.

2. Calculate these values for the pan band in the same way as in (10)-(15), only first separate the even frames from the odd frames and calculate two values for both a_0 and R_0^2.

a. $A_{even}(s,f) \approx a_{0,even}(s)A_{VRP,even}(s,f),\ f = 2,4,6,...$ (16)

$A_{odd}(s,f) \approx a_{0,odd}(s)A_{VRP,odd}(s,f),\ f = 2,4,6,...$ (17)

b. $A_{estimate,even}(s,f) = a_{0,even}(s)A_{VRP,even}(s,f) + C_{lr,even},\ f = 2,4,6,...$ (18)

$$SS_{err,even}(b,s)=\sum_{f=1}^{N_f}\left(A_{estimate,even}(b,s,f)-A_{even}(b,s,f)\right)^2,\ f=2,4,6,... \tag{19}$$

$$\overline{A}_{even}(b,s)=\frac{1}{N_f}\sum_{f=1}^{N_f}A_{even}(b,s,f),\ f=2,4,6,... \tag{20}$$

$$SS_{tot,even}(b,s)=\sum_{f=1}^{N_f}\left(A_{even}(b,s,f)-\overline{A}_{even}(b,s)\right)^2,\ f=2,4,6,... \tag{21}$$

$$R_0{}^2{}_{even}(s)=1-\frac{SS_{err,even}(b,s)}{SS_{tot,even}(b,s)} \tag{22}$$

$$A_{estimate,odd}(s,f)=a_{0,odd}(s)A_{VRP,odd}(s,f)+C_{lr,odd},\ f=1,3,5,... \tag{23}$$

$$SS_{err,odd}(b,s)=\sum_{f=1}^{N_f}\left(A_{estimate,odd}(b,s,f)-A_{odd}(b,s,f)\right)^2,\ f=1,3,5,... \tag{24}$$

$$\overline{A}_{odd}(b,s)=\frac{1}{N_f}\sum_{f=1}^{N_f}A_{odd}(b,s,f),\ f=1,3,5,... \tag{25}$$

$$SS_{tot,odd}(b,s)=\sum_{f=1}^{N_f}\left(A_{odd}(b,s,f)-\overline{A}_{odd}(b,s)\right)^2,\ f=1,3,5,... \tag{26}$$

$$R_0{}^2{}_{odd}(s)=1-\frac{SS_{err,odd}(b,s)}{SS_{tot,odd}(b,s)} \tag{27}$$

Section 3.

1. For each band (except for the Pan band), each SCA and each detector
   a. Using a least squares fit calculate a coefficient a_1 and constant C_1 that relates the means over N_1 collects to the per-SCA VRP means from the same SCA as that detector over the same N_1 collects.

$$S(b,s,d,c)\approx a_1(b,s,d)\overline{A_{VRP}}(b,s,c)+C_1(b,s,d) \tag{28}$$

   b. Calculate the R-squared values from the fit, R_1^2

$$S_{estimate}(b,s,d,c)=a_1(b,s,d)\overline{A_{VRP}}(b,s,c)+C_1(b,s,d) \tag{29}$$

$$SS_{err}(b,s,d)=\sum_{c=1}^{N_c}\left(S_{estimate}(b,s,d,c)-S(b,s,d,c)\right)^2 \tag{30}$$

$$\overline{S}(b,s,d)=\frac{1}{N_c}\sum_{c=1}^{N_c}S(b,s,d,c) \tag{31}$$

$$SS_{tot}(b,s,d)=\sum_{c=1}^{N_c}\left(S(b,s,d,c)-\overline{S}(b,s,d)\right)^2 \tag{32}$$

$$R_1^{\,2}(b,s,d) = 1 - \frac{SS_{err}(b,s,d)}{SS_{tot}(b,s,d)} \quad (33)$$

2. Calculate these values for the pan band in the same way as in (28)-(33) only first separate the even frame means from the odd frame means and calculate two values for each a_1, C_1, and R_1^2.

   a. $$S_{even}(b,s,d,c) \approx a_{1,even}(b,s,d)\overline{A_{VRP,even}}(b,s,c) + C_{,even1}(b,s,d) \quad (34)$$

   $$S_{odd}(b,s,d,c) \approx a_{1,odd}(b,s,d)\overline{A_{VRP,odd}}(b,s,c) + C_{1,odd}(b,s,d) \quad (35)$$

   b. $$S_{estimate,even}(b,s,d,c) = a_{1,even}(b,s,d)\overline{A_{VRP,even}}(b,s,c) + C_{1,even}(b,s,d) \quad (36)$$

   $$SS_{err,even}(b,s,d) = \sum_{c=1}^{N_c}\left(S_{estimate,even}(b,s,d,c) - S_{even}(b,s,d,c)\right)^2 \quad (37)$$

   $$\overline{S}_{even}(b,s,d) = \frac{1}{N_c}\sum_{c=1}^{N_c} S_{even}(b,s,d,c) \quad (38)$$

   $$SS_{tot,even}(b,s,d) = \sum_{c=1}^{N_c}\left(S_{even}(b,s,d,c) - \overline{S}_{even}(b,s,d)\right)^2 \quad (39)$$

   $$R_{1,even}^{\,2}(b,s,d) = 1 - \frac{SS_{err,even}(b,s,d)}{SS_{tot,even}(b,s,d)} \quad (40)$$

   $$S_{estimate,odd}(b,s,d,c) = a_{1,odd}(b,s,d)\overline{A_{VRP,odd}}(b,s,c) + C_{1,odd}(b,s,d) \quad (41)$$

   $$SS_{err,odd}(b,s,d) = \sum_{c=1}^{N_c}\left(S_{estimate,odd}(b,s,d,c) - S_{odd}(b,s,d,c)\right)^2 \quad (42)$$

   $$\overline{S}_{odd}(b,s,d) = \frac{1}{N_c}\sum_{c=1}^{N_c} S_{odd}(b,s,d,c) \quad (43)$$

   $$SS_{tot,odd}(b,s,d) = \sum_{c=1}^{N_c}\left(S_{odd}(b,s,d,c) - \overline{S}_{odd}(b,s,d)\right)^2 \quad (44)$$

   $$R_{1,odd}^{\,2}(b,s,d) = 1 - \frac{SS_{err,odd}(b,s,d)}{SS_{tot,odd}(b,s,d)} \quad (45)$$

3. Write a_0, $a_{0,even}$, $a_{0,odd}$, a_1, $a_{1,even}$, $a_{1,odd}$, $C_{1,even}$, and $C_{1,odd}$, to the BPF. Write R_0^2, $R_0^2{}_{,even}$, $R_0^2{}_{,odd}$, R_1^2, $R_1^2{}_{,even}$, and $R_1^2{}_{,odd}$ to the characterization database.

4. If selected, write the BPF to a file

   Note that for any pixel marked as inoperable, dropped, or having been affected by impulse noise or saturation will be excluded from all calculations.

7.5.1.6 Maturity

Level-3. No heritage exists in either Landsat or ALI processing for this algorithm.

In step 6, 7, and 8 some number of lines may need to be skipped in order to eliminate any transients from the beginning of the shutter collect. This was done with some of the SWIR bands for ALIAS processing.

Note that only pre-acquisition shutter data is used in the main algorithm where coefficients are calculated. From the EDU test data we've concluded that we can get away with only one shutter collect and there isn't that much difference, and the pre-acquisition collect is probably easier to use in this algorithm. However, we may decide later that we need both.

There may be enough similarity between using the pre- and post- shutter acquisition averages that we may be able to not use the post shutter acquisition averages.

The coefficients a_0 that are calculated may be stable enough from collect to collect that it may be possible to calculate them only once for use on all images.

A tolerance may be added on the integration time check.

It may be better to make Section 3 its own algorithm apart from sections 1 and 2 if the coefficients calculated there must be calculated more often than the coefficients calculated in Section 2.

7.5.2 OLI Bias Determination

7.5.2.1 Background

Removing the detector's bias from each detector's data is a necessary first step in the conversion from the raw detector signal to radiance (and reflectance) as part of product generation. The bias determination algorithm estimates the bias to remove from each pixel. There are several ways in which the bias can be estimated. One way uses a per-detector bias estimated by averaging shutter data on a per detector basis. Another way uses a pair of coefficients a_1 and C_1 to relate the average per-SCA VRP response to the average per-imaging detector dark response. These coefficients are then applied to the per-SCA VRP response to estimate the average per-detector dark response. Frame to frame variation in the bias can also be estimated by using the VRP data and a coefficient a_0 that relates the per-frame behavior of the VRPs to the per frame behavior of the detector dark response within a scene.

The selection of bias model parameters used to derive the bias to be applied is specified via parameters/flags set in the processing work order. These flags are defined as options below.

7.5.2.2 Input

Description	Level	Source	Type
VRP cross track averages corresponding to the scene (A_{VRP})	N_{Bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Calculated from image data	Float

Description	Level	Source	Type
Dropped frames in VRP data	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Mask	Integer
Impulse noise in VRP data	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Mask	Integer
Saturated pixels in VRP data	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Mask	Integer
VRP operability list	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	CPF	Integer
Pre-acquisition shutter average (S_a)	N_{bands} x N_{SCAs} x $N_{detectors}$	BPF	Float
Post-acquisition shutter average (S_b)	N_{bands} x N_{SCAs} x $N_{detectors}$	BPF	Float
Bias model parameter (a_0)	N_{bands} x N_{SCAs}	BPF	Float
Bias model parameter (a_1)	N_{bands} x N_{SCAs} x $N_{detectors}$	BPF	Float
Bias model parameter (C_1)	N_{bands} x N_{SCAs} x $N_{detectors}$	BPF	Float
CPF Bias (b_{CPF})	N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Float

Output

Description	Level	Target	Type
Bias	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	Bias Removal or file	Float

7.5.2.3 Options

- Output bias values to file (default off)
- bias selection
    - Per-detector bias (no estimate of per frame variation included)

1. Pre-acquisition shutter average (S_a)
2. Post-acquisition shutter average (S_b)
3. Average of pre- and post-acquisition shutter averages (S_{ab}, default)
4. CPF bias (b_{CPF})
5. Estimate using a_1, C_1, and the per-SCA VRP averages

- Per-frame bias (estimate of per frame variation included)
  - Source of a_0, a_1, and C_1
    1. Current BPF
    2. Selected date/time for alternate BPF
  - Five options for calculating C
    1. Pre-acquisition shutter average (S_a)
    2. Post-acquisition shutter average (S_b)
    3. Average of pre- and post-acquisition shutter averages (S_{ab}, default for the per frame bias)
    4. CPF bias (b_{CPF})
    5. Estimate using a_1, C_1, and the per-SCA VRP averages

7.5.2.4 Procedure

If a per-detector bias is selected where the option number is included in 1-4, then retrieve the selected bias values from the BPF. If option 3 has been selected, then calculate S_{AB}, the average of S_A and S_B.

$$S_{AB}(b,s,d) = \frac{1}{2}\left(S_A(b,s,d) + S_B(b,s,d)\right) \quad (1)$$

where d is detector, s is SCA, and b is band.

If a per-detector bias using option 5 is selected, then retrieve a_1 and C_1 from the BPF. Calculate the per-detector bias estimate with these parameters.

For each band excluding the pan band

- For each SCA
  - Find the cross-track average of all of the VRP detectors.

    $$A_{VRP}(b,s,f) = \frac{1}{N_d}\sum_{d=1}^{N_d} Q_{VRP}(b,s,d,f) \quad (1)$$

    where f is line, Q_{VRP} is the VRP detector response for the specific band, N_d is the number of VRP detectors in a band, and A_{VRP} is the VRP average for frame f. Note that any anomalous pixels will be excluded from calculations.
  - Calculate the per-SCA average of the VRP data across all frames.

    $$\overline{A_{VRP}}(b,s) = \frac{1}{N}\sum_{f=1}^{N} A_{VRP}(b,s,f) \quad (2)$$

  - For each detector
    - Calculate the per-detector estimate of the bias mean.

      $$E_{avg}(b,s,d) = a_1(b,s,d)\overline{A_{VRP}}(b,s) + C_1(b,s,d) \quad (3)$$

      Where E_{avg} is the estimate of the bias average.
- For the pan band, follow the procedure for the other bands, only treat the odd and even frames separately. Note again that all anomalous pixels are excluded from calculations.

$$A_{VRP,even}(b,s,f)=\frac{1}{N_d}\sum_{d=1}^{N_d}Q_{VRP}(b,s,d,f),\ f=2,4,6,... \quad (3)$$

$$A_{VRP,odd}(b,s,f)=\frac{1}{N_d}\sum_{d=1}^{N_d}Q_{VRP}(b,s,d,f),\ f=1,3,5,... \quad (5)$$

$$\overline{A_{VRP,even}}(b,s)=\frac{1}{N}\sum_{f=1}^{N}A_{VRP,even}(b,s,f),\ f=2,4,6,... \quad (4)$$

$$\overline{A_{VRP,odd}}(b,s)=\frac{1}{N}\sum_{f=1}^{N}A_{VRP,odd}(b,s,f),\ f=1,3,5,... \quad (5)$$

$$E_{avg,even}(b,s,d)=a_{1,even}(b,s,d)\overline{A_{VRP,even}}(b,s)+C_{1,even}(b,s,d) \quad (6)$$

$$E_{avg,odd}(b,s,d)=a_{1,odd}(b,s,d)\overline{A_{VRP,odd}}(b,s)+C_{1,odd}(b,s,d) \quad (7)$$

- Write the per-SCA VRP cross track averages to the database.

The result of any of these options should be an array containing a single value for every detector of every band, with the exception being the Pan band where for every detector there is an odd frame estimate and an even frame estimate. Expand this array such that there is one bias value for every frame of the image to be corrected, except for in the case of the Pan band where the even frame estimate should be expanded to only the number of even frames, and the odd frame estimates expanded to only then number of odd frames. The result will be a single value for every pixel in the image to be corrected where every frame belonging to the same detector is identical. **Note** that 1.5 DN should have been added to non-barrel-shifted data during convert to float. This is to account for the average error due subtracting the lower 12 of 14-bit data from the upper 12 of 14-bit data.

If the selected option is to have the bias include an estimate of the per-frame variation, calculate the per-frame bias based on bias model parameters from the specified BPF.

- For each band excluding the pan band
  - For each SCA
    - Retrieve the bias model parameters (a_0) from the specified BPF.
    - Calculate the per-SCA average of the VRP data across all frames.

      $$\overline{A_{VRP}}(b,s)=\frac{1}{N}\sum_{f=1}^{N}A_{VRP}(b,s,f) \quad (8)$$

      where $\overline{A_{VRP}}$ is the VRP data average and *N* is the number of frames.

  - For each SCA, detector, and frame
    - If one of the options 1-4 for calculating *C* is selected
      - Retrieve the values of S, where S is per detector bias values from the source selected from the previous list.
      - Calculate the constant value *C*. $C(b,s,d)=S(b,s,d)-a_o(b,s)\overline{A_{VRP}}(b,s)$ (9) where *S* is the selected per detector bias. Note that no pixels marked as bad or anomalous are included in this calculation.
    - Otherwise, if option 5 is chosen, then *C* is calculated in exactly the same way as in (9) except S is $E_{avg,}$ as written in (10). $C(b,s,d)=E_{avg}(b,s,d)-a_o(b,s)\overline{A_{VRP}}(b,s)$ (10)

- Calculate the per frame bias estimate.
  $$bias(b,s,d,f) = a_0(b,s)A_{VRP}(b,s,f) + C(b,s,d) \quad (11)$$ where *bias* is the bias estimate.

- For the pan band, follow the procedure for the other bands, only treat the odd and the even frames separately.

$$\overline{A_{VRP,even}}(b,s) = \frac{1}{N_{even}} \sum_{f=1}^{N_{even}} A_{VRP,even}(b,s,f), f = 2,4,6,... \quad (12)$$

$$\overline{A_{VRP,odd}}(b,s) = \frac{1}{N_{odd}} \sum_{f=1}^{N_{odd}} A_{VRP,odd}(b,s,f), f = 1,3,5,... \quad (13)$$

$$C_{even}(b,s,d) = S_{even}(b,s,d) - a_{o,even}(b,s)\overline{A_{VRP,even}}(b,s) \quad (14)$$

$$C_{odd}(b,s,d) = S_{odd}(b,s,d) - a_{o,odd}(b,s)\overline{A_{VRP,odd}}(b,s) \quad (15)$$

$$bias_{even}(b,s,d,f) = a_{0,even}(b,s)A_{VRP,even}(b,s,f) + C_{even}(b,s,d) \quad (16)$$

$$bias_{odd}(b,s,d,f) = a_{0,odd}(b,s)A_{VRP,odd}(b,s,f) + C_{odd}(b,s,d) \quad (17)$$

Note that dropped frames will be excluded from calculations and will be set to zero in *bias*.

Send the bias to Bias Removal, and if the option is selected, to file.

7.5.2.5 Maturity

Level 3, although portions of this algorithm originated from ALIAS and the L7 IAS before that, the per-frame bias calculation is new.

The coefficients a_0, a_1, and C_1 may be stable enough from collect to collect that it would be possible to calculate it less often than every interval.

There may be enough similarity between using the pre- and post- shutter acquisition averages that we may be able to not use the post shutter acquisition averages.

Instead of a simple average of pre- and post-acquisition shutter averages, an interpolation function could be used so that either a per-frame or per-scene bias would be used. This would only occur if there was a significant drift in the pre-to-post-acquisition shutter averages.

7.5.3 OLI Bias Removal

7.5.3.1 Background

Conversion to radiance (L1R) occurs in 3 steps: bias removal; response linearization; and gain (absolute and relative) application. This algorithm addresses the first step in generating the L1R radiance product, removing bias. Applying gain and linearizing the detector response are addressed in separate algorithms. Bias removal is accomplished by subtracting a value (in DN) from each pixel of the input image. This value varies by detector for all bands, and also by frame.

Options to select between a CPF bias and biases derived from shutter data acquired near the collect are available for special processing. An option for bias temperature sensitivity correction (described in a separate algorithm) is applied within this algorithm.

Input

Descriptions	Level	Source	Type
Scene (L0R)	$N_{bands,}$ x N_{SCAs} x $N_{Detectors}$ x N_{Lines}	L0R	Float
Per line correction	$N_{bands,}$ x N_{SCAs} x $N_{Detectors}$ x N_{Lines}	Bias Determination	Float
Per detector correction*	$(N_{bands,}+1)$x N_{SCAs} x $N_{Detectors}$	CPF	Float
Temperature Correction Factor (CF_T – Unitless)	$N_{bands,}$ x N_{SCAs} x $N_{Detectors}$	Temperature Sensitivity Correction	Float

*(N_{bands}+1) accounts for the pan band being separated into odd and even detectors

Output

Descriptions	Level	Target	Type
Bias Corrected Scene	$N_{bands,}$ x N_{SCAs} x $N_{Detectors}$ x N_{Lines}	Response Linearization	Float
Choice of Bias	1	L1R Metadata	String
Temperature Sensitivity Correction Flag	1	L1R Metadata	Integer

7.5.3.2 Options

- Apply temperature sensitivity correction (default off)
- Choice of bias
  - From CPF
  - From bias determination algorithm (default)

7.5.3.3 Procedure

For each band, SCA, detector and line

1. If temperature sensitivity correction is selected, multiply the temperature correction factor CF_T by the corresponding bias.

$$bias' = CF_T(b,s,d) \cdot bias(b,s,d,f) \quad (1)$$

   where *b* is band, *s* is SCA, *d* is detector, and *f* is line, and if the bias is to come from the CPF, then *f* in (1) is one.

2. Subtract the per-line or per-detector bias from the corresponding input scene pixel. If temperature sensitivity correction is selected, use (2). Otherwise, use (3).

$$Q'(b,s,d,f) = Q(b,s,d,f) - b'(b,s,d,f) \quad (2)$$

$$Q'(b,s,d,f) = Q(b,s,d,f) - b(b,s,d,f) \quad (3)$$

   where *Q* is the input scene data *Q'* is the output bias corrected scene data, and if the bias is to come from the CPF, then *f* in *b* and *b'* in (2) and (3) is one.

7.5.3.4 Maturity

Level 2 (ALIAS reuse)
The only difference between this algorithm and the algorithm used in ALIAS is the per-line biases. Depending on OLI test data, this may or may not be needed operationally. If per-line biases aren't needed, this algorithm will be simplified by removing the line variables in equation (1) and (2).

The temperature correction factor may also be changed to be additive, or possibly both a multiplicative and additive term may be needed.

7.5.4 OLI Characterize Radiometric Stability (16-day)

7.5.4.1 Background

Radiometric stability of an instrument is fundamental to low uncertainty in the radiometric calibration of data products generated from its data. OLI has requirements on its band average stability over several time intervals, specifically 60 second (about 2 scenes), up to 16-day (one repeat cycle) and 16-days up to 5 years (mission lifetime). This algorithm specifically addresses the 16 day radiometric stability as given as:

OLI-1001 For Bands 1-8, over any time period up to 16 days, after radiometric correction per 5.3.1.2, with one set of gain coefficients that were determined prior to the 16 day period, the scene averaged OLI image data for radiometrically constant targets with radiances greater than or equal to L-typical shall not vary by more than plus or minus 1% (95% or 2 sigma confidence interval) of measured radiance.

OLI-1522 For Band 9, over any time period up to 16 days, after radiometric correction per 5.3.1.2, with one set of gain coefficients that were determined prior to the 16 day period, the scene averaged OLI image data for radiometrically constant targets with radiances greater than or equal to Ltypical shall not vary by more than plus or minus 2% (95% or 2 sigma confidence interval) of measured radiance

After launch the 16-day Radiometric Stability will be used for OLI Key Performance Requirement (KPR) verification.

Radiometric Stability KPR:
The key performance metric is the variation in the band average response of the instrument to a constant radiance (greater than or equal to $L_{typical}$) over any period of time up to 16 days. Of the measurements made over these periods, 95% need to be within 1.2% of the average value for all bands but the cirrus band.

This algorithm characterizes the stability of the OLI bands radiometric response using the on-board calibration devices. In particular, the working Stim lamps will be used every day and the working solar diffuser will be used nominally every 8 days. This algorithm, though not directly related to requirements could also be run on shutter data to characterize dark response stability.

This algorithm will run on bias-corrected and linearized digital numbers from the Lamp Response Characterization and/or Histogram Statistics Characterization database tables; they do not require separate analysis of image data.

7.5.4.2 Dependencies

Histogram Statistics Characterization
Lamp Characterization
Diffuser Characterization

7.5.4.3 Inputs

The inputs to this algorithm come from either the output of other algorithms (DB) or from a set of calibration parameters (CPF). Table 1 lists the inputs of this algorithm.

Table 1: Algorithm Inputs

Descriptions	Symbol	Units	Level	Source	Type
Lamp Acquisition info: date, time, ID				DB	
Number of samples		Counts		DB	Int
Level-1 Statistics (bias-corrected, linearized)	Q_i, σ_i	DN	N_{band} X N_{SCA} x N_{det}	DB (lamp response table)	Float
Diffuser acquisition info: date, time, ID				DB	
Number of diffuser samples		Count		DB	
Earth-sun distance	d	[]		JPL model	float
Level-1 Statistics (bias-corrected, linearized)	Q_i, σ_i	DN	N_{band} X N_{SCA} x N_{det}	DB (solar diffuser table)	Float
Inoperable detectors, out-of-spec detectors				CPF	integer
Relative Gains	r_{CPF}	[]	N_{band} X N_{SCA} x N_{det}	CPF	float
Moving window size (multiple window sizes possible, different sizes for lamp and diffuser)	W	Days			integer

7.5.4.4 Output

Descriptions	Symbol	Units	Level	Source	Type
Time interval samples (multiple arrays depending on window sizes)					
Window size		Days			
Number of samples	$N_{samples}$				
Traveling average means	$\bar{x}$, s_i	DN	N_{band} X N_{SCA} x N_{det} x $N_{samples}$		Float

Traveling average uncertainties	CV, unc_{CV}	[]	N_{band} X N_{SCA} x N_{det} x $N_{samples}$		
Time periods where the band-average traveling average does not meet spec					

7.5.4.5 Options

Report data to ascii report files as well as IDL save file and plots.

Prototype Code
Stability.pro
Traveling_average.pro
Ploterror.pro, oploterror.pro
Fpm_legend.pro, leg.pro
Tvread.pro

7.5.4.6 Procedure

For each appropriate collect, for each band:

1. Extract Lamp Characterization and Diffuser Characterization database table data for each detector; Q_i, σ_i, N_{valid}.
    a. For the lamp, only extract the working lamp data, since the other bulb pairs won't be used often enough to provide meaningful trends.
    b. For the panel, use only the working panel data
2. Calculate FPM-average means from per-detector means
3. Calculate band-average means from per-detector means, Q_{BA}
4. For diffuser data, correct the band-average signal for the earth-sun distance
    a. $Q = Q_{BA} * d^2$
5. For each traveling average window (lamp data, use window sizes of 6, 12 and 16 days; panel, use window size of 16 days) for each per-detector, fpm average and band average:
    a. Calculate traveling average mean and stdev: $\bar{x}$ and s.
    b. [Secondary test for KPR] check if two times the CV minus two times the absolute uncertainty in the CV is greater than the specification value. This is a less stringent test than the Chi^2 test.
        i. Calculate uncertainty in stdev in each sample in the traveling average: $\frac{unc_s}{|s|} = \frac{1}{\sqrt{2*(n-1)}}$ (this is the relative uncertainty in s)
        ii. Calculate uncertainty in mean for each sample in the traveling average: $unc_{\bar{x}} = \frac{s}{\sqrt{n}}$ (this is the absolute uncertainty in $\bar{x}$)
        $\frac{unc_{\bar{x}}}{|\bar{x}|} = \frac{s}{\bar{x}\sqrt{n}}$ (this is the relative uncertainty in $\bar{x}$)
        iii. Calculate Coefficient of Variation (CV) and the uncertainty in the CV for each sample in the traveling average
        $CV = \frac{s}{\bar{x}}$

$$\frac{unc_{CV}}{|CV|} = \sqrt{\left(\frac{unc_s}{|s|}\right)^2 + \left(\frac{unc_{\bar{x}}}{|\bar{x}|}\right)^2} = \sqrt{\frac{1}{2\,(n-1)} + \frac{s^2}{\bar{x}^2 n}}$$

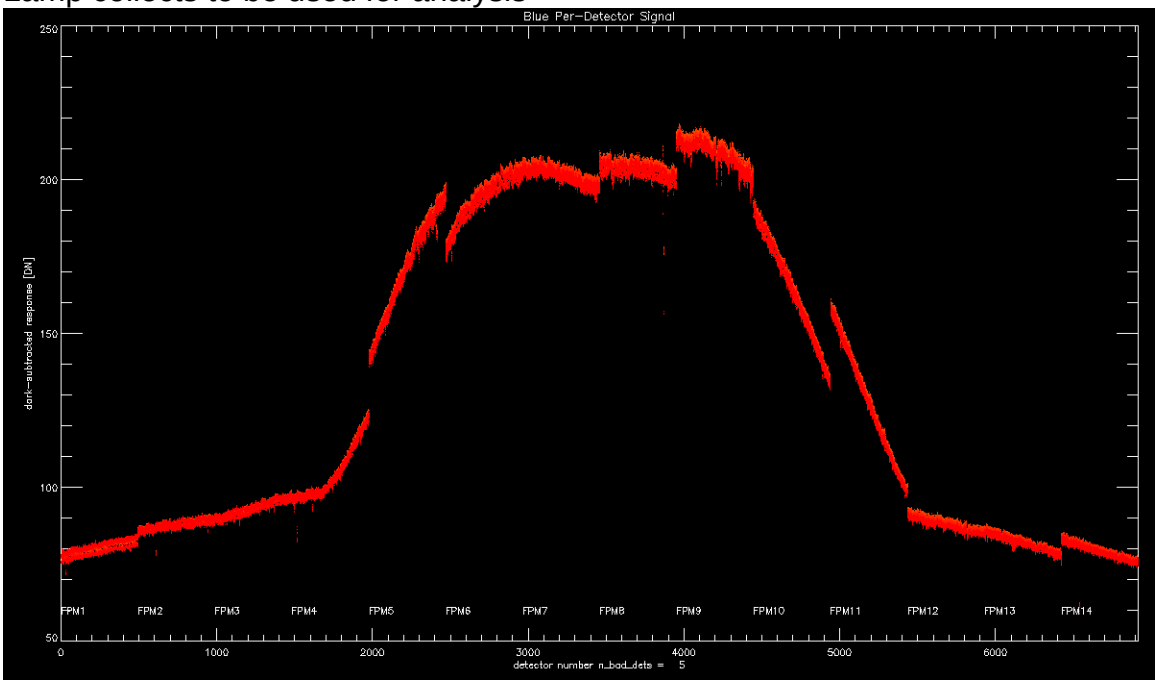
c. [Primary test for KPR] Calculate Chi^2 to test the hypothesis that the variance is less than the specification value.
   i. H_0: s^2 <= specification, H_A: s^2 > specification
   ii. $\nu = n - 1$, $\sigma_0^2 = \left(\frac{spec}{2.0}\bar{x}\right)^2$, $\alpha = 0.05$
   iii. $\chi^2 = \frac{\nu s^2}{\sigma_0^2}$
   iv. $\chi^2_{\alpha,\nu}$= chisqr_cvf(0.05, ν) [IDL command for Chi^2 95% value.]
   v. H_0 passes if $\chi^2 < \chi^2_{\alpha,\nu}$ and the band-average is not out-of-spec.

6. For the band-average results, flag bands where the Chi^2 H_0 is rejected. Mark periods as out-of-specification based on the Chi^2 test; decide whether to flag the band as out-of-specification.

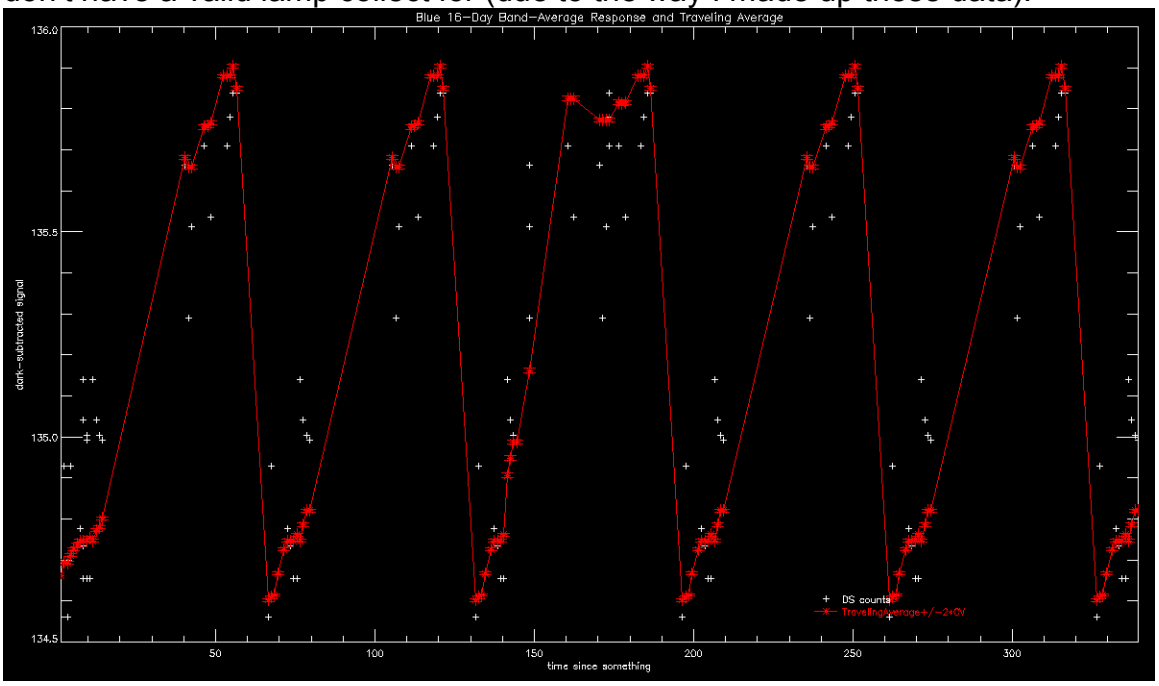
7.5.4.7 Maturity

The code is meant for the IAS tool box and thus, is not robust and hands-off as are the usual IAS algorithms. I expect that I will watch the output pop-up as I run this on a monthly basis and will add and subtract functionality as I see fit.

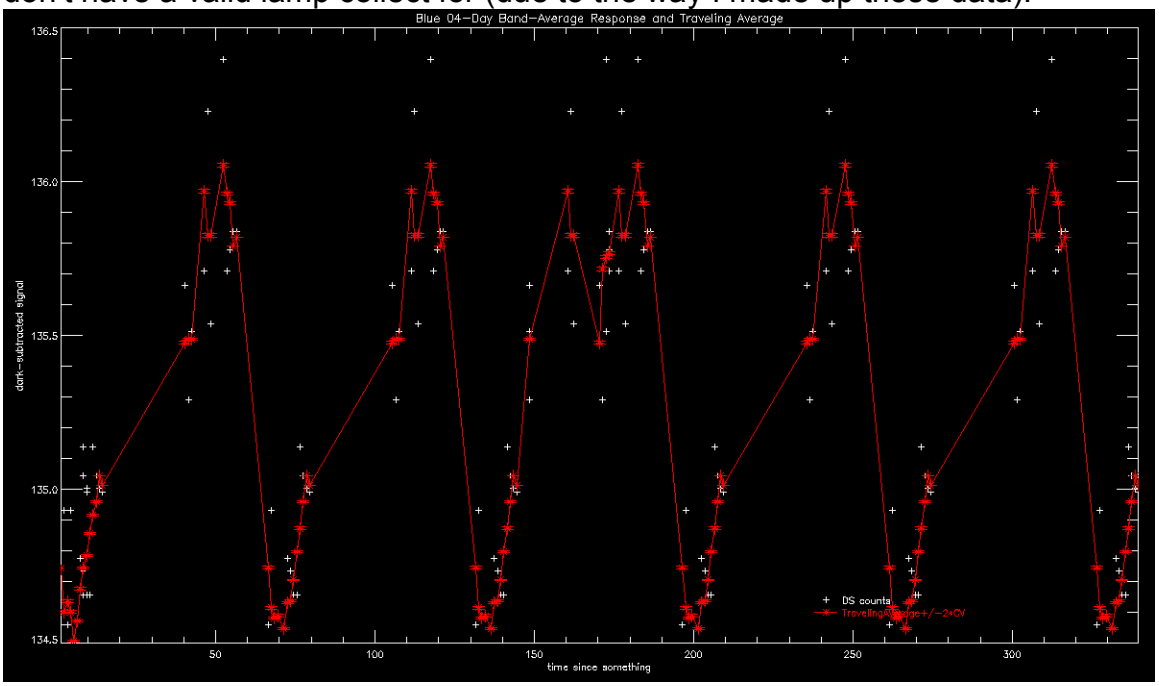
Sample plots:
Lamp collects to be used for analysis



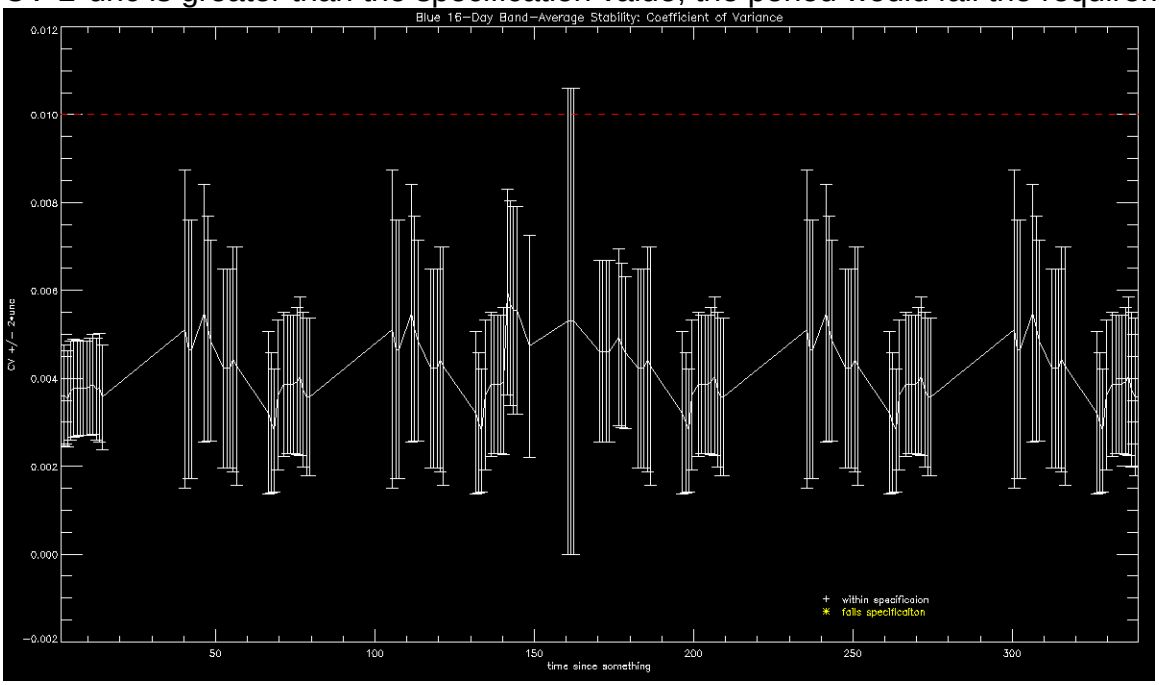
Lamp 16-day traveling average for the band- average signal. There are some 16 day periods where I don't have a valid lamp collect for (due to the way I made up these data).



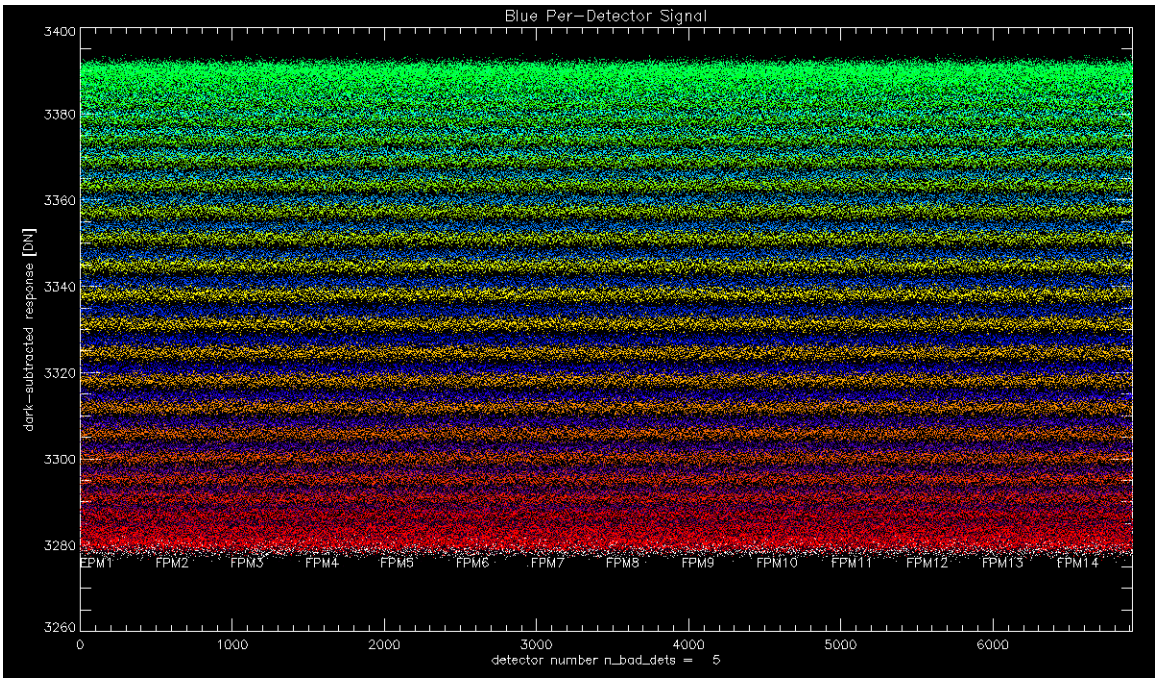
Lamp 4-day traveling average for the band-average signal. There are some 4 day periods where I don't have a valid lamp collect for (due to the way I made up these data).



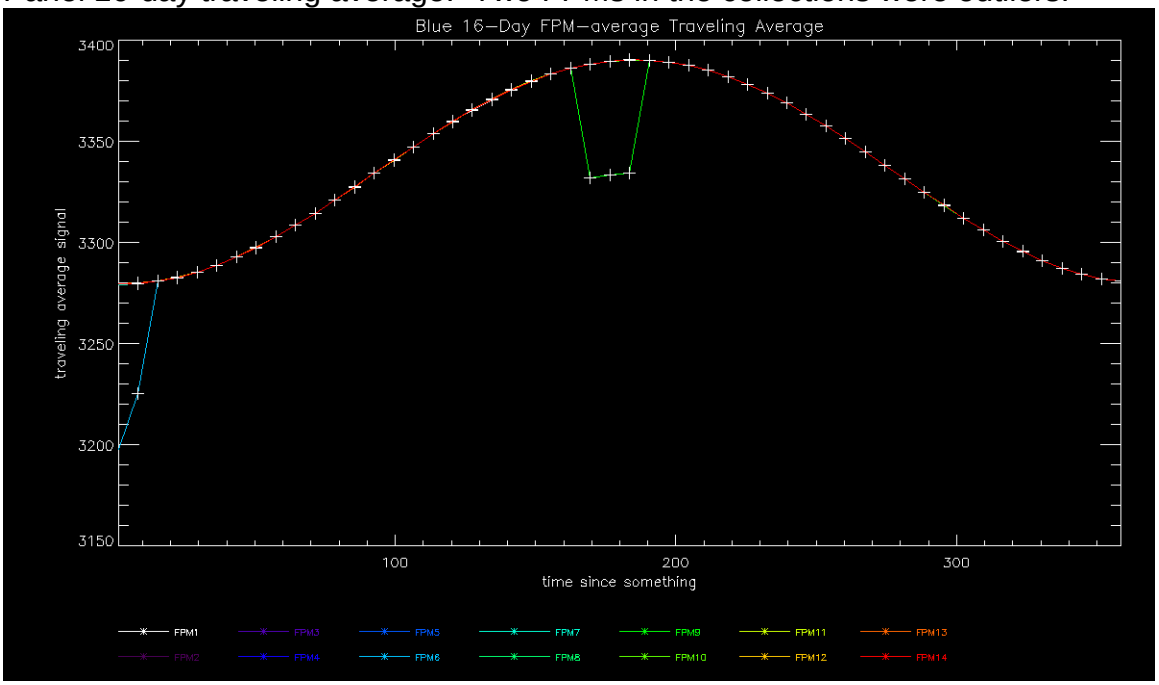
CV of 16-day lamp band-average traveling average, with requirement indicated by dashed line. If CV-2*unc is greater than the specification value, the period would fail the requirement.



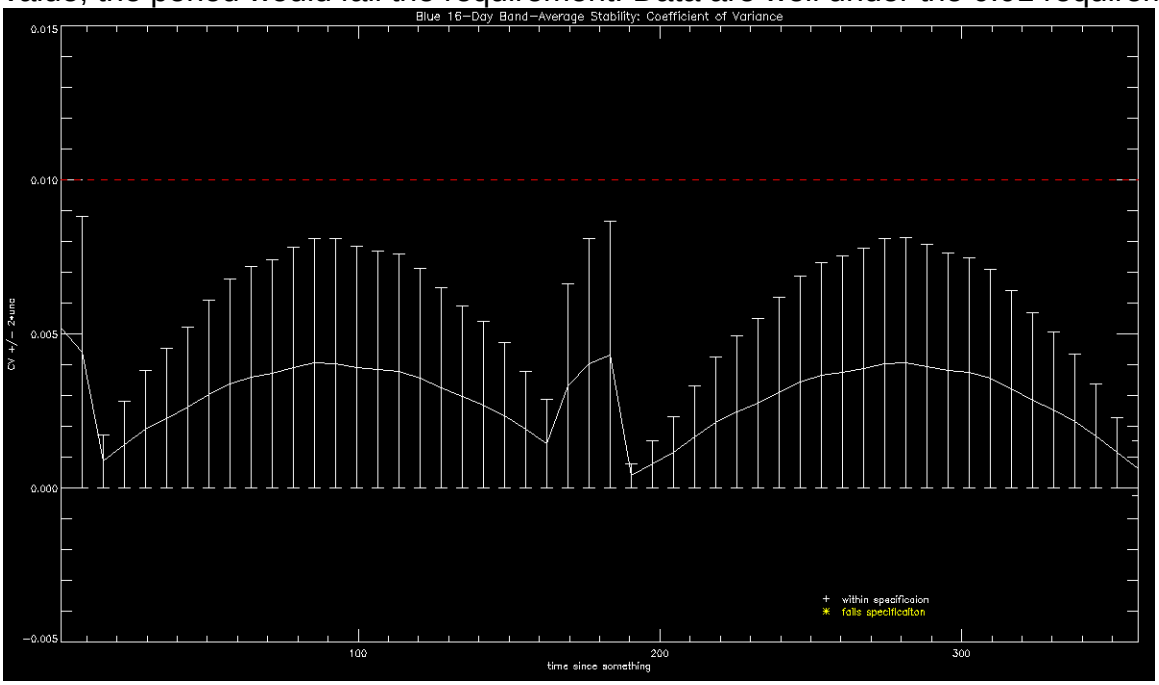
Panel data to be used in analysis (all fabricated)



Panel 16-day traveling average. Two FPMs in the collections were outliers.



CV of 16-day panel band-average traveling average. If CV-2*unc is greater than the specification value, the period would fail the requirement. Data are well under the 0.01 requirement.



Earth-Sun Distance Calculation

The JPL Ephemeris table (DE421) describes the orbits of the sun and planets with very high precision over relatively long time scales[8]. The file is stored as a series of Chebyshev coefficients which can be interpolated to essentially any desired temporal accuracy. The IDL tool that I use to read from the JPL file requires the date as input and outputs a three element position array. To convert to earth-sun distance in AU:

```
;need month, day, year to
ydn2md, year, doy, month, day
pos = run_ephem( month, day, year )
dist = double( sqrt( pos(0)^2 + pos(1)^2 + pos(2)^2. ) )
earth_sun_distance = dist / aukm       ;e-s dist correction
```

7.5.5 OLI Nonlinear Response Characterization (OLI)

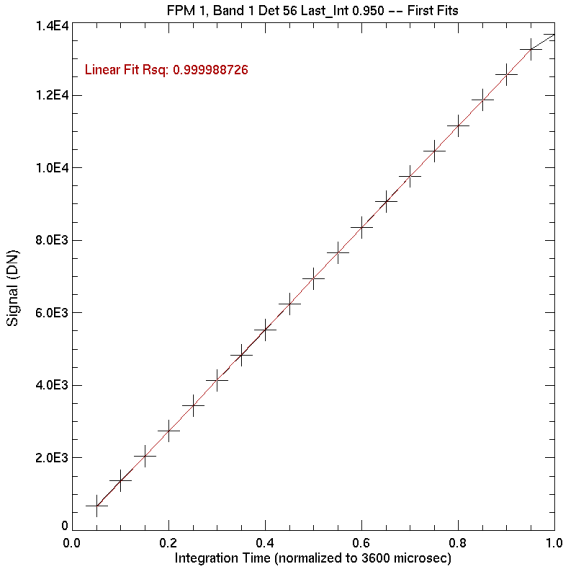
7.5.5.1 Background

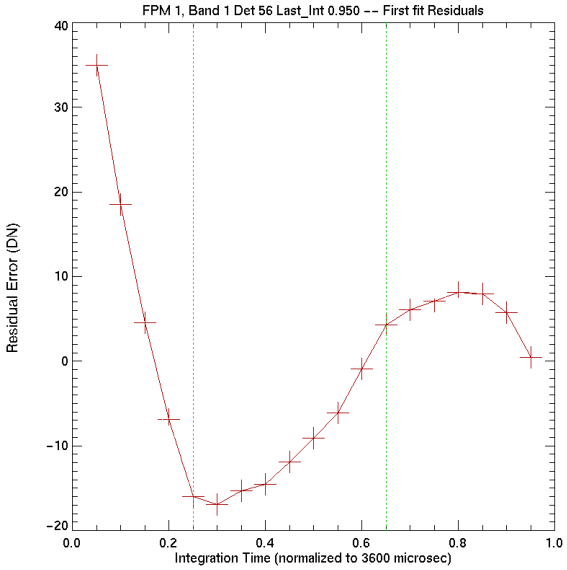
The output of the OLI instrument is quantized output (Q) in units of digital number (DN). This Q is expected to be related to the input signal of the detectors, but that relationship may not be linear.

[8] http://lheawww.gsfc.nasa.gov/users/craigm/bary/

Each detector may have unique non-linear irregularities in response that must be corrected in processing.

Figure 1 shows a response slope for a typical detector from Band 1, SCA 1. The integration time is linearly related to input radiance and is normalized to the radiance setting of the integrating sphere (DSS) used in the Ball Aerospace radiometric test collections. It can be seen that the detector response is very linear within its dynamic range. It is expected that non-linear behavior occurs just before the high and low saturation levels. There is some non-linear behavior, however, even in the center of the response -- Figure 2 shows a plot of the residuals of a linear fit made within the detector's dynamic range. All detectors studied exhibit this type of behavior.





From test collections made either prelaunch or in orbit, a set of parameters can be derived to linearize the detector response. The preferred data should be Integration Time Sweep (ITS) collections made with the diffuser panel as a background.

The intended form of the linearization equation is piecewise quadratic, with three distinct regions. The cutoff points between the regions – the points where the functions intersect – are determined by equating the adjacent functions. The first (bottom) region extends from zero up to the first minimum of the linear fit residual plot. The second (middle) region extends up to the last experimental point within the detector's dynamic range. The third (top) region covers the most non-linear portion of the detector response – from the top of the detector's dynamic range to the high analog saturation point.

Inputs

Descriptions	Symbol	Units	Level	Source	Type
L0 (Bias corrected) Mean for each integration time	Q	DN	N_{band} x N_{det} x N_{levels}	Db (Histogram)	Float

collection					
Radiance level for each integration time collection	R	W/m^2 sr µm	N_{levels}	Db (Histogram)	Float
Integration time for each collection	i		N_{levels}		Float

7.5.5.2 Outputs

Descriptions	Symbol	Units	Level	Target	Type
Remapping function cutoff thresholds		DN	N_{scas} x N_{bands} x $N_{detectors}$ x $N_{cutoffs}$	CPF	Float
Remapping function parameters			N_{scas} x N_{bands} x $N_{detectors}$ x $N_{coefficients}$ x ($N_{cutoffs}$+1)	CPF	Float
Mean absolute residual		DN	N_{scas} x N_{bands} x $N_{detectors}$	Report file	Float
Maximum residual		DN	N_{scas} x N_{bands} x $N_{detectors}$	Report file	Float

Note: $N_{cutoffs}$ equals 2 in the current implementation.

7.5.5.3 Options

The cutoff point table below (in Procedure section 2.1) contains an array of floating point values that are input to the work order but that should be editable by the operator.

7.5.5.4 Procedure

1.0 Prepare Diffuser Data

Each integration time sweep collection for OLI will involve a Diffuser collection taken at an integration time setting, which is given in the metadata. For each collection the mean DN value for each detector should be stored in the database by the Histogram Statistics procedure.

To prepare data for linearity characterization, the diffuser collections to use should be identified and the per-detector means obtained. For each collection the solar angle and viewed radiance for each band should be calculated. Once collated, the prepared data should be an array of [Q,R,i] ([mean DN, viewed radiance, integration time]) for every band and detector.

2.0 Linearity Characterization

2.1 Find preliminary cutoff points.

The cutoffs are initially chosen as the available effective radiances that are nearest to the radiances from the following table:

Band	Band name	First cutoff R (W/m^2 sr μm)	Second cutoff R (W/m^2 sr μm)
1	CA	147.25	265.05
2	B	121.00	211.75
3	G	113.61	198.81
4	R	96.01	192.02
5	NIR	67.87	135.74
6	1SW	14.90	26.08
7	2SW	7.36	11.04
8	P	195.16	362.44
9	CRS	23.01	64.43

The effective radiance of each integration time sweep collection is calculated:

$R(i) = R_{max}*i$

The collection whose effective radiance is nearest to the first cutoff point is labeled i_{low}:

$Q_{low} = Q(d, i_{low})$
$R_{low} = R(i_{low})$

The collection with effective radiance nearest to the second cutoff point is labeled Q_{high} (with corresponding R_{high} and i_{high}).

These labels are made once for each band. These 'a priori' radiance cutoff levels should be work order parameters that are adjustable by the IAS analysts.

2.2 Create ideal line

Once the cutoff points have been found, the ideal line for each detector can be calculated. On OLI, linear behavior is assumed to pass through the origin (0 DN, 0 Radiance). The slope of the ideal line is then:

$M(d) = Q_{high}/ R_{high} = Q(d, i_{high})/(R_{max}*i_{high})$

An ideal line curve is then created, with one point for every integration time:

$Q_{ideal}(d,i) = M(d)*R_{max}*i$

2.3 Polynomial fit

The parameters of the linearization are then found by fitting the actual data to the ideal line. This is done in three regions: 0 to i_{low}, i_{low} to i_{high}, and i_{high} to the highest available effective radiance.

The fit is currently done with the IDL routine POLY_FIT, which uses the matrix inversion method that can be found in Numerical Recipes. Any implementation of a two-dimensional polynomial fit can be used. The output of each fit for each region should be quadratic coefficients such that:

$$Q'(d,r,i) = c(r,2)*Q(d,r,i)^2 + c(r,1)*Q(d,r,i) + c(r,0) \approx Q_{ideal}(d,r,i)$$

where Q'(d,r,i) = The linearized DN values (one for every i).
c(r,n) = The *nth* order quadratic coefficient for region *r*.

2.4 Determine actual cutoff points.

The actual cutoff points should be calculated after the quadratic coefficients, so that the fit between regions is as smooth as possible. The cutoff points are calculated by finding the point at which the fitted curves for two regions meet. This point can be calculated with the quadratic equation:

$$Q'_{low} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

where a = The difference in the second-order coefficients for regions 1 and 2.
a = c(2,2) – c(1,2)
b = The difference in the first-order coefficients for regions 1 and 2.
b = c(2,1) – c(1,1)
c = The difference in the zeroth-order coefficients for regions 1 and 2.
a = c(2,0) – c(1,0)

The quadratic equation has two solutions, due to the ± in the equation, and neither solution is guaranteed to be a real or positive number. Both solutions should be calculated, and the real component compared to the 'a priori' cutoff point. The solution that is closest to the first 'a priori' cutoff value Q_{low} should be chosen as the final cutoff value.

Similarly, Q'_{high} is calculated between the second and third regions (c(3,n) – c(2,n)), and its two solutions are compared to the second 'a priori' cutoff, Q_{high}.

2.5 Report output.

After the parameters have been created, the data should be linearized using them and residuals to the ideal line should be calculated. This array of residuals is used by the IAS analyst to check the accuracy of the linear characterization.

$$Residuals(d,i) = Q'(d,i) – Q_{ideal}(d,i)$$

The mean and maximum absolute residual for each detector should be reported in an output file.

7.5.5.5 Maturity

Possible changes to this algorithm are:

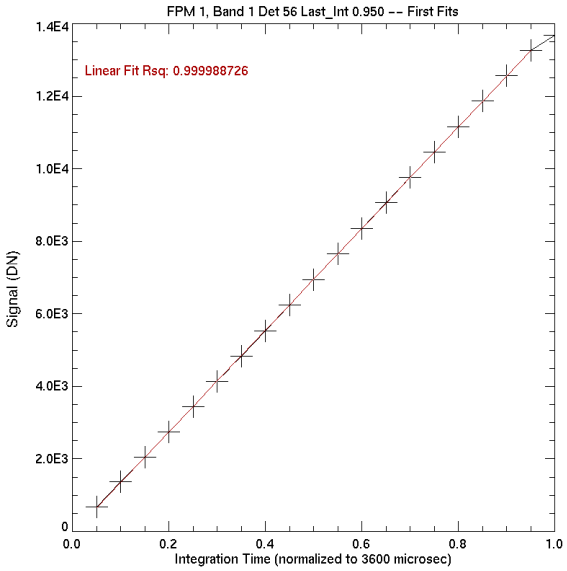
- **Change in Algorithm** – The linearity correction algorithm could change, which would necessitate a change to the linearity characterization.

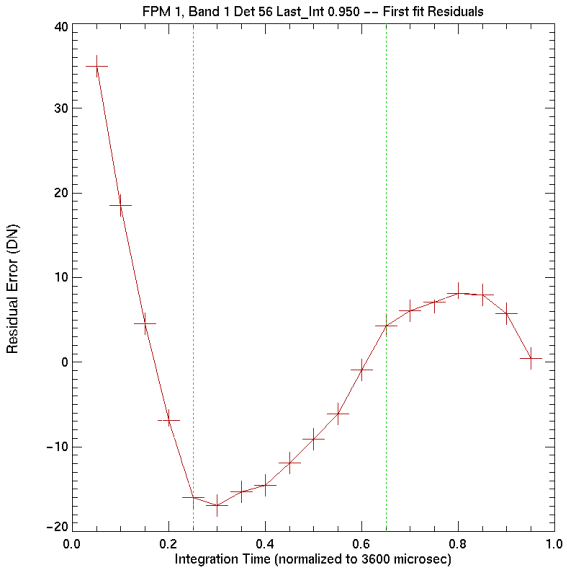
7.5.6 OLI Response Linearization

7.5.6.1 Background/Introduction

The output of the OLI instrument is quantized output (Q) in units of digital number (DN). This Q is expected to be related to the input signal of the detectors, but that relationship may not be linear. Each detector may have unique non-linear irregularities in response that must be corrected in processing.

Figure 1 shows a response slope for a typical detector from Band 1, SCA 1. The integration time is linearly related to input radiance and is normalized to the radiance setting of the integrating sphere (DSS) used in the Ball Aerospace radiometric test collections. It can be seen that the detector response is very linear within its dynamic range. It is expected that non-linear behavior occurs just before the high and low saturation levels. There is some non-linear behavior, however, even in the center of the response -- Figure 2 shows a plot of the residuals of a linear fit made within the detector's dynamic range. All detectors studied exhibit this type of behavior.





From the Ball test collections, a set of parameters can be derived to linearize the detector response. The intended form of the linearization equation is piecewise quadratic, with three distinct regions. The cutoff points between the regions – the points where the functions intersect – are determined by equating the adjacent functions.

The first (bottom) region extends from zero up to the first minimum of the linear fit residual plot. The second (middle) region extends up to the last experimental point within the detector's dynamic range. The third (top) region covers the most non-linear portion of the detector response – from the top of the detector's dynamic range to the high analog saturation point.

There are two limitations in the current calibration dataset. There are not enough data points at the bottom and middle of the detector response to fit a quadratic function, so the square term for those regions will be set to zero. Also, the top region lacks enough data to actually see non-linear behavior. This will give us a poor polynomial fit for the top region. Future pre-flight calibration collections will address this issue. Data in this region is not expected to be seen during flight; the detector dynamic ranges are designed to encompass the entirety of Earth-based target radiances.

7.5.6.2 Input

Description	Level	Source	Type
Scene (L0R)	N_{scas} x N_{bands} x $N_{detectors}$ x N_{lines}		Float
Remapping function cutoff thresholds	N_{scas} x N_{bands} x $N_{detectors}$ x $N_{cutoffs}$	CPF	Float
Remapping function parameters	N_{scas} x N_{bands} x $N_{detectors}$ x $N_{coefficients}$ x ($N_{cutoffs}$+1)	CPF	Float

The planned implementation is for two cutoff points ($N_{cutoffs}$ = 2) and three sets of quadratic coefficients ($N_{coefficients}$ = 3).

7.5.6.3 Output

Description	Level	Target	Type
Scene (linearized)	N_{scas} x N_{bands} x $N_{detectors}$ x N_{lines}		Float

7.5.6.4 Procedure

Response linearization should be a simple replacement of the incoming value with a linearized response value.

The linearized value will be calculated for each sample point by an algorithm. These algorithms will be determined by pre-launch calibration of the detectors. As described above, this algorithm is a multi-segment quadratic function with the detector Q value as its only variable. Parameters for the linearization function will be stored in the CPF. For this method the required parameters are three sets of quadratic coefficients (one set for each sloped segment), and two cutoff points.

1.1. For each input frame
1.2. For each band, SCA, and detector
  1.2.1. Compare input Q with cutoff values and select appropriate quadratic coefficients.

    The input Q value is compared to the cutoff thresholds in the CPF, to determine which segment of the linearization to use.
    If input is less than cutoff #1, then segment 1 is used.
    If input is greater than or equal to cutoff #1 but less than cutoff #2, then segment 2 is used.
    If input is greater than or equal to cutoff #2, then segment 3 is used.

  1.2.2. Evaluate the quadratic function at the input Q:

    $output = qp[0, s] + qp[1, s] * input + qp[2, s] * (input)^2$

    where input = the input value, Q, in DN.
    output = the output value, Q_1, in linearized DN.
    qp[x, s] = linearization parameter x, for segment s.
    These parameters come from the CPF.

  1.2.3. Return the output value.

7.5.6.5 Maturity

A process to analyze the on-orbit integration time sweep data and produce linearization Calibration Parameters will be discussed and produced at some later time.

Currently CPF parameters exist only for a few FPM-Band combinations. Expect to have a more extensive test data set in the future.

More finely resolved test data for linearization may be generated for the flight array sometime in 2011 when the OLI pre-flight testing occurs.

Other possible changes that may occur are:

- **Lookup Table** – The remapping could also be implemented as a look-up table (LUT), which will return the “linearized” value (Q_1) for each instrument Q value. This table will be different for each detector and will return a floating-point number. The LUT is populated from the re-

mapping function for each detector. This can be pre-calculated and stored as part of the calibration, or can be calculated at run time. This method may be quicker than calculating the re-mapping function as the curve fit functions and cut-off points don't have to be evaluated for each sample. It also allows a different function (other than quadratic) to be used to fit the linearization data. The LUT could also be generated with manual adjustments and delivered at part of a calibration input if necessary. A drawback is that a large floating point LUT (up to 4096 elements) for each detector must be held in memory during processing.

- **Change in Algorithm –** The current linearization algorithm has been adapted from the TIRS linearization algorithm as suitable for OLI. A different algorithm may be specified by Ball Aerospace, which will force us to adopt it.

7.5.7 OLI Alternate Response Linearization (OLI)

7.5.7.1 Background

The OLI detectors will have a non-linear relationship between the radiance viewed and the DN value output by the detectors. Because of this, the response of the detectors must be linearized as part of radiometric calibration.

The correction for non-linearity involves remapping bias corrected detector response data onto a linear curve. In the alternate response linearization algorithm, this is done in two stages; one linearity correction, and one per-detector non-uniformity correction (NUC). In practice these corrections are similar and use the same algorithm (but different lookup tables (LUTs)), making linearity correction a two-step process.

The standard response linearization procedure uses a quadratic equation, but the instrument vendors have delivered correction parameters as full-sized lookup tables that cover all possible values in 14-bit space. The alternate response linearization algorithm is intended as an implementation of the vendor algorithms, with the sole difference that the very large vendor-provided LUTs have been compressed for performance reasons into
piecewise continuous arrays that contain paired input and output values. The piecewise continuous LUT provides the same results as the full-sized LUT to within 0.001 DN, but take up approximately one-thirtieth of the disk space.

This algorithm will be used in the IAS as an occasional check to verify that the ground system is capable of linearizing the detectors using the parameters provided by the vendors. It is not intended for LPGS use or for use on every scene.

The Alternate Response Linearization procedure will also be run on TIRS data. LUTs for TIRS will be generated but do not yet exist. The algorithm will be exactly the same; only the size of the arrays will differ.

7.5.7.2 Input

Description	Symbol	Units	Level	Source	Type

Scene (L0rc2 - bias corrected)	*Q*	DN	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float
Artifact Mask	*AM*	Unitless	N_{bands} x N_{SCAs}		Int
Linearity Correction LUT	*LIN*	Unitless	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{values}	RLUT file (an extension of the CPF)	Float
Uniformity Correction LUT	*NUC*	Unitless	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{values}	RLUT file (an extension of the CPF)	Float

7.5.7.3 Output

Description	Symbol	Units	Level	Target	Type
Scene (L0rc2 - bias corrected, linearized)	*Q**	DN*	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Float

7.5.7.4 Options

- Apply Linearity correction (default on)
- Apply Non-uniformity correction (default on)

7.5.7.5 Procedure

Response linearization should be a simple replacement of the incoming value with a linearized response value.

For each band there are two LUTs that are condensed versions of the LUTs provided by Ball. Each LUT has a pair of values – an index value and an output adjustment value – for every SCA and detector. The first LUT is intended to correct for SCA-wide nonlinearities. Ball refers to this as the 'linearity' correction (abbreviated as LIN). The second LUT is intended to correct per-detector nonlinearities. Ball refers to this as the 'nonuniformity' correction, or NUC. Despite their distinct names, both are currently per-detector corrections using the exact same algorithm.

Whenever alternate response linearization is performed, both the LIN and NUC corrections are run. The options to turn off one or the other correction exist only as a means to debug the RLUTs.

A search is made in INPUT LUT to find the index in the list whose values bracket the incoming DN value. Those bracketing indices are then used to get the bracketing values from the VAL LUT. The adjustment is a scaled interpolation between those two bracketing adjustment values. The output value (DN*) is the input DN plus the calculated adjustment.

The paired LUTs are:

INPUT[band, sca, det, index]
VAL[band, sca, det, index]

Although these LUTs are dependent upon band, sca, and det, they are referred to as INPUT[index] and VAL[index] below for clarity. Note also that the size of the LUTs varies between detectors. The INPUT and VAL array are always the same size as each other, but the arrays used by one detector may not be the same size as the arrays used by any other detector. The maximum size of these arrays is fixed at 1000 by the compression algorithm (described below).

With

index = The largest index into the INPUT array where INPUT[index] ≤ DN.

(The values INPUT[index] and INPUT[index+1] should bracket the incoming DN value. Finding this index will entail a quick search through the INPUT array.)

Then the adjustment is an interpolation between the LUT values:

Adj = *VAL[index] + (VAL[index+1] - VAL[index]) * (DN - INPUT[index]) / (INPUT[index+1] - INPUT[index])*

*DN** = *DN + Adj*

This process should be used for the LIN LUT and then the NUC LUT, for every pixel in an image.

Note that because index+1 is used in the algorithm, pixels whose values equal the 14-bit integer maximum (16383) should not be processed. This is far beyond saturation in most bands and should not be an issue.

The specific algorithm that compresses the vendor-provided LUTs into the INPUT and VAL LUTs is not important to the alternate response linearization algorithm, but it does dictate the size of the LUTs. Any compression algorithm may be used, as long as it results in LUTs of the same format and with the same fidelity to the original LUTs.

To clarify , the procedure for the compression algorithm is as follows:

For each detector in each band, the first non-zero adjustment in the vendor LUT (VLUT) is found as index i_0. Between the indices i_0 and i_0+1, the adjustment changes by a delta adjustment (ΔA). The linear equation describing the adjustment is then:

$$Adj(i) = (i - i_0) * \Delta A + VLUT(i_0)$$

The index *i0* is placed in the INPUT LUT, and the VLUT adjustment at that index is placed into the VAL LUT.

This equation is checked against the VLUT value at subsequent indices (*i+1*, *i+2*...) until the difference between this extrapolated adjustment and the actual VLUT adjustment is greater than the threshold value (currently set at 0.001 DN). When the threshold is exceeded, the linear extrapolation is then reset; a new i_0 is set at the index where the threshold was exceeded and becomes the next entry in the INPUT LUT, while the VLUT adjustment at the new i0 becomes the next entry in the VAL LUT. Then a new delta is calculated, and the iterations resume with the new extrapolation.

At the end of the process, a final point is placed into the arrays; the index 16,383 and the associated adjustment value at that index. This insures that the linearization algorithm will always find two indices to bracket any incoming DN value.

This compression algorithm is run for both the LIN and NUC LUTs provided by the vendor, for every fpm, band, and detector. The output becomes the RLUTs used by the alternate response linearization algorithm.

7.5.7.6 Maturity

Level 2 (No Landsat or ALI reuse)

Possible complications or changes that may be made in the response linearization function are:

- **Simplified INPUT array** – A change to the LUT compression algorithm could be made so that the compressed LUTs are no longer of variable length. This could simplify the INPUT array, making the index search through it easier, or possibly obsolete the INPUT array entirely and allow the index to be a linear function of the incoming DN. Whether this is possible or not, and what the trade off in size will be, is yet to be demonstrated.
- **Unpacked INPUT array** – To improve performance, the INPUT LUT (whatever its form) could be unpacked in memory before processing into a straight lookup array, with one value per integer value from 0 to 16,383. This would improve performance without increasing the size of the RLUT files, but might introduce memory issues.
- **Double Per-Detector Lookup** – Currently Ball's algorithm describes one SCA-wide correction and one per-detector NUC, but the 'SCA-wide' correction is filled as a per-detector array. This capability should be retained to match Ball's algorithm. If Ball's algorithm changes so that the SCA LUT is truly independent of the detectors then the LDCM version should change to match that.

7.5.8 OLI Detector Response Characterization (Solar Diffuser)

7.5.8.1 Background

The Primary radiometric calibration devices on OLI are solar diffusers. These Spectralon panels are rotated in position in front of the OLI aperture and the spacecraft is oriented so as to bring sunlight in the solar diffuser port so that sunlight is reflected into the OLI aperture. There are 2 solar diffusers on-board OLI, a Primary and pristine. The current plan calls for the Primary diffuser to be deployed approximately weekly during normal operations. The pristine diffuser will be deployed on a less frequent basis and used to monitor degradation of the Primary panel. The OLI diffuser data provides a valuable source for deriving radiometric gain updates for L1r processing, performing instrument

characterizations and monitoring uniformity and stability requirements. Both radiance and reflectance based gains will be derived and stored.

To derive radiance gains, both bias and nonlinearity corrected OLI data (0Rc) values (in DN) and diffuser spectral radiance values (W/m2-sr □m) are required. With the exception of a correction for Earth-Sun distance, the diffuser spectral radiance values are relatively static as the sun reflects off the diffuser at a fixed angle and the diffusers are expected to change slowly in reflectance characteristics. The radiance values adjusted to a reference earth-sun distance can be stored in the CPF. (Note. there may be multiple versions of these values i.e. pre-launch, post-launch and current that could be stored in a single CPF or as separate CPFs). The Earth-Sun distance correction will be derived from JPL ephemeris data. Gains will be generated and output as per detector relative gains, and band-averaged gains. Similar outputs for reflectance-based gains will be derived based on spectral reflectance values (unitless) in place of the spectral radiance values.

Approximately 500 frames of data with the solar diffuser illuminated will be acquired per typical collect. However, there are some less frequent acquisitions that will change the size of these data sets. These collects will be acquired at variable integration times for non-linearity characterization and only the nominal integration time portion of these collects will be processed by this algorithm. A second acquisition scenario involves 60 second long solar diffuser collects; these collects are used to evaluate the short-term stability of the OLI response.

Ephemeris and attitude data will be processed and used to confirm the solar pointing during each solar acquisition. This processing may occur in part separately prior to use in this algorithm. It is assumed that all ancillary data for verification of the acquisition will come from the wideband ancillary data. Separately, the MOC will provide the CVT a report on the solar calibration maneuver indicating whether the maneuver was executed as planned, e.g., the solar pointing was as required.

To track pristine and Primary diffuser datasets a diffuser parameter needs to be identified, associated with the appropriate inputs and output parameters and stored in the trending DB. Artifacts will be accounted for in the L0rc inputs to processing.

Assessment of algorithm outputs will be performed by comparing the “original” solar radiance and reflectance calibrated products, to “new” solar products derived with the newly derived gains. While not performed by this algorithm, these product generation steps should be considered a routine part of “solar characterization process”. The evaluation of these products and reports will be performed ‘off-line’

Inputs

Descriptions	Symbol	Units	Level	Source	Type
L0rc1 (Bias corrected			Nband	Db	Float

and linearized) Means	Qnet	DN	x Ndet	(Histogram)	
L0rc Stdev1	Qσnet	DN	Nband x Ndet	Db (Histogram)	Float
Start time		GMT		Db	Float
Stop time		GMT		Db	Float
Number of lines used				Db (Histogram)	Int
Inoperable Detectors			Nband x Ndet	CPF	Int
Attitude Control System Quaternion Coefficients	Q0,Q1,Q 2,Q3			Ancillary Preprocessin g	Fltarr
Diffuser Radiance2 @nominal SZA, E-S Distance (1A.U.)	L□	W/m2 sr □m	NDT x Nband x Ndet	CPF	Float
Diffuser Deployment Angle		degrees		Ancillary File	Float
Earth –Sun Distance 3	des	AU		Auxiliary File	Float
Diffuser Bidirectional Reflectance Factor2	ρ		NDT x Nband x Ndet	CPF or Ancillary	Float
Integration Time				Header or DB	
Diffuser Type (DT) i.e. Pristine or Primary				L0R	
Start_time				L0R	
Stop_time				L0R	
Collection Type				L0r, Metadata	
Sun Zenith Angle Mean	Θs	degrees		Report, Db	
Sun Zenith Angle Stdev	Θsσ	degrees		Report, Db	Float
Sun Azimuthal Angle Mean	Φs	degrees		Report, Db	Float
Sun Azimuthal Angle Stdev	Φsσ	degrees		Report, Db	Float

Notes:

[1] Mask artifacts presumed to be accounted for in the imagery, prior to generating histogram statistics. Histogram statistics should also include the solar integration time sweep data acquired at the nominal integration time.

[2] Parameters must be stored separately identifiable by the Diffuser Type (DT i.e. Pristine or Primary).

[3] From JPL data

7.5.8.2 Outputs

Descriptions	Symbol	Units	Level	Target	Type
Start time				Report, Db	Float
Stop time				Report, Db	Float
Lines averaged				Report, Db	Int
Relative Gains Mean [2]	G_{rel}		N_{band} x N_{det}	Db	Float
Relative Gain Stdev[2]	$G\sigma_{rel}$		N_{band} x N_{det}	Db	Float
Radiance-Band Avg Gains Mean [2]	G	DN/ W-m^2-sr-μm	N_{band}	Report, Db	Float
Radiance-Band Avg Gains Stdev [2]	$G\sigma$	DN	N_{band}	Report, Db	Float
Reflectance-Band Avg Gains Mean [2]	$G\rho$	DN	N_{band}	Report, Db	Float
Reflectance-Band Avg Gains Stdev [2]	$G\rho\sigma$	DN	N_{band}	Report, Db	Float
Diffuser Type	DT			Report, Db	Int
Deployment Angle		degrees		Report, Db	Float
View Sun Zenith Angle Mean[3]	Θ_v	degrees		Report, Db	Float
View Sun Zenith Angle Stdev[3]	$\Theta_v\sigma$	degrees	N_{band}	Report, Db	Float
View Sun Relative Azimuthal Angle Mean[3]	Φ_v	degrees	N_{band}	Report, Db	Float
View Sun Relative Azimuthal Angle Stdev[3]	$\Phi_v\sigma$	degrees		Report, Db	Float
Incident Sun Zenith Angle[3]	Θ_s	degrees		Report,Db	Float
Incident Sun Zenith Angle Stdev[3]	Θ_s	degrees		Report,Db	Float

7.5.8.3 Options

Summary report to be generated with every solar processing run.

7.5.8.4 Procedure

Note, while all procedure steps will be identical for both Diffuser Types (DT) Primary panel (both nominal and Time Sweep acquisitions) and the Pristine panel, certain inputs (as identified by the input table) needs to be identified/processed by its Diffuser Type (DT).

1.0 Verify solar acquisition:

1.1 Read in ancillary data and verify/output solar collect information e.g.
Diffuser Deployment Resolver Position
Collect Sequence

Integration Time
Verify the integration time is nominal.

IF not verified THEN stop processing ELSE

1.2 Derive Solar Angles:

Sun angles are calculated in the LDCM Solar Calibration Frame of Reference system (LSCF). In the solar calibration mode the spacecraft is aligned to the LSCF i.e. the pitch axis is aligned to the sun (-Y), and the yaw axis to the projected nadir vector (+Z). *Angle(s) should be constant over the entire acquisition with pointing control..* Fig 1 illustrates the vectors and angles wrt to a diffuser panel (small oval), mounted within the rotating calibration assembly (large oval).

θs = incident angle between diffuser normal and sun vector direction
θv = view (scatter) angle between diffuser normal and sensor line-of sight vector.

Both theta angles should be about 45 degrees wrt prelaunch alignment of the diffuser prelaunch that should remain constant throughout the mission.

φv = view (scatter) relative azimuthal angle between diffuser normal and sensor line-of sight

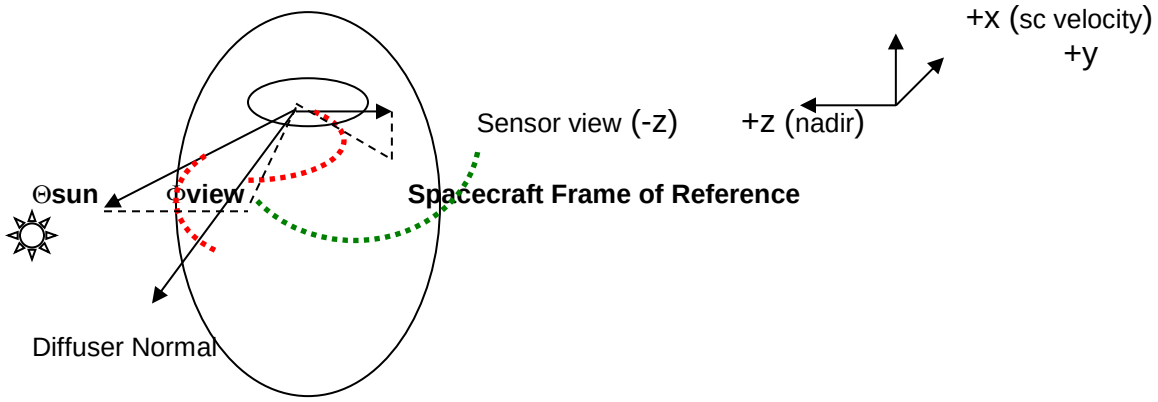


Fig 1 Diffuser Solar Angles Defined wrt Diffuser Normal Component

1.2.1 Process S/C Attitude: Extract spacecraft (ACS) quaternion coefficients from ancillary data preprocessing, corresponding to calibration interval.

1.2.2 Retrieve JPL Solar Ephemeris i.e. sx,sy,sz

For all Attitude quaternion values in solar cal acquisition

1.2.3 Transform sun vectors from earth-centered to LSCF for all ephemeris points in acquisition.

Xs,Ys,Zs = TR * sx,sy,sz using values from 1.2.1 and 1.2.2

Where:

TRansform Matrix =
TR(0,0) = 1-2*(q2(i)^2+q3(i)^2)
TR(1,0) = 2*(q1(i)*q2(i)+q0(i)*q3(i))
TR(2,0) = 2*(q1(i)*q3(i)-q0(i)*q2(i))
TR(0,1) = 2*(q1(i)*q2(i)-q0(i)*q3(i))
TR(1,1) = 1-2*(q1(i)^2+q3(i)^2)
TR(2,1) = 2*(q0(i)*q1(i)+q2(i)*q3(i))
TR(0,2) = 2*(q0(i)*q2(i)+q1(i)*q3(i))
TR(1,2) = 2*(q2(i)*q3(i)-q0(i)*q1(i))
TR(2,2) = 1-2*(q1(i)^2+q2(i)^2)

1.2.4 Derive array of sun angles using transformed sun components i.e.
Xs, Ys, Zs,

panel orientation components i.e.
Xn = 0.
Yn = -cos(45.)
Zn = -sin(45.)

and view direction components i.e.
Xv = 0.
Yv = 0.
Zv = 1

Incident Solar Zenith Angle

cosi = Xn*Xs + Yn*Ys+Zn*Zs
sini = sqrt (1-cosi^2)
ϑs= atan(sini/cosi)

View Solar Zenith Angle

cosi = atan (Xn*Xv + Yn*Yv+Zn*Zv)
sini = sqrt (1-cosi^2)
ϑv = atan(sini/cosi)

View Solar Relative Azimuthal Angle

cosa = Xns*Xnv+Yns*Ynv+Zns*Znv
sina = sqrt(1-cosa^2)
φv = atan (sina/cosa)

Where Xns,Yns,Zns = sun normal , the cross product of XnYnZn with XsYs,Zs i.e.

Xnv,Ynv,Znv = view normal , the cross product of XnYnZn with XYZ-view direction components i.e. the only non-zero view component (zv=1) along the sensor view axis.

End Attitude loop

1.2.5 Derive sun angle means and standard deviations

Θs = mean(ϑs) & $\Theta s\sigma$ = Stdev(ϑs)
Θv = mean(ϑv) & $\Theta v\sigma$ = Stdev(ϑv)
Φv =mean(φv) & $\Phi v\sigma$ = Stdev(φv)

7.5.8.4.1 2.0 DERIVE RADIANCE GAINS:

FOR each Band

2.1 Read per detector spectral radiances - L_λ (b,d)

2.2 For date of acquisition, read in appropriate value of Earth-Sun distance, d_{es}

FOR each detector

2.3 Read in L0rc means ($<Q_{net}(b,d)>$) and standard deviations ($Q\sigma_{net}(b,d)$) derived from histogram statistics

2.4 Derive absolute per detector gain mean and standard deviation, i.e.

$$G(b,d) = d_{es}^{2} * <Q_{net}(b,d)> / L_\lambda(b,d) \quad (3)$$
$$G\sigma(b,d) = d_{es}^{2} * Q\sigma_{net}(b,d) / L_\lambda(b,d) \quad (3a)$$

END detector loop

2.5. Generate/output the band mean gain and standard deviation (over all operable detectors)

$$G_{aBands}(b) = < G(b,*)> \ \& \ G\sigma_{aBands}(b) = \text{Stdev}(G(b,*)) \quad (4)$$

2.6 Generate/output the per detector mean relative gain and standard deviation , wrt to the per band average.

$$G_{rel}(b,d) = G(b,d) / G_{aBands}(b) \ \&$$
$$G\sigma_{rel}(b,d) = G\sigma(b,d) / G_{aBands}(b) \quad (5)$$

END band loop

7.5.8.4.2 3.0 DERIVE REFLECTANCE GAINS:

For each bands

For each detector:

3.1 Read per detector bidirectional reflectance factors $\rho(b,d)$

3.2 Derive absolute per detector average reflectance gains $G\rho(b,d)$, and per detector reflectance gain standard deviations $G\rho\sigma(b,d)$ = i.e.

$$G\rho(b,d) = (Q_{net}(b,d) * d_{es}^{2}) / (\rho(b,d) * \cos(\Theta_s)) \quad (6)$$
$$G\rho\sigma(b,d) = (Q\sigma_{net}(b,d) * d_{es}^{2}) / (\rho(b,d) * \cos(\Theta s)) \quad (6a)$$

Where $Q_{net}(b,d)$ = per detector diffuser corrected response

$\rho(b,d)$ = diffuser reflectance factors
Θ_s = solar zenith diffuser angle (nominally 45°)
d_{es} = Earth-Sun distance

End detector loop

3.3 Generate/output the band mean gain and standard deviation (over all operable detectors)

$G\rho_{aBands}(b) = < G\rho(b,*)>$ and $G\sigma_{aBands}(b) = \text{Stdev}\ (G\rho(b,*))$ (7)

End bands

4.0 Generate summary report for every processing run with hardcoded formats. (Example format)

SOLAR CALIBRATION SUMMARY REPORT

Report Date
Start Acq Time
Stop Acq Time
Total Acq Time

Diffuser Type: (i.e. Pristine, Primary)
Collect Sequence Type: (i.e. Nominal, Integration Time Sweep, etc)
Integration Time:

Solar Distance and Diffuser Angles:
- E-S Distance (AU)
- Deployment Angle (degs)
- Solar Zenith Angle (Theta-i, degs)
- View Zenith Angle (Theta-v, degs)
- Sun Azimuth Angle (Phi-v, degs)

Ephem Outliers:

Image Stats (Bias Corrected & Linearized)
BandID Avg Stdev

Diffuser Radiance & Diffuser Gains
BandID Avg Avg Stdev

Diffuser Reflectance & Reflectance Gains
BandID Avg Avg Stdev

Total Frames Processed:

7.5.8.5 Maturity

- This algorithm can be adapted to process actual scene data for alternative relative gain derivation method.

Issues:

- Assessments of CPF spectral radiance values based upon prelaunch measurements, could lead to in-line derivation of radiance for each acquisition.
- Hardcoded report format.

7.5.9 OLI Standalone 60 Second Radiometric Stability Characterization

7.5.9.1 Background/Introduction

Both the bias and the gain instrument stability of an instrument are contributing factors to variability within a radiometrically calibrated product. The OLI has a "60 second stability", a.k.a, short-term stability requirement specifically designed to control this within product variability. The specific requirement states: "Over any time period up to 60 seconds, after radiometric correction per 5.3.1.2, the scene-averaged OLI image data for radiometrically constant targets with radiances greater than or equal to L-typical shall not vary by more than plus or minus 0.5% (95% or 2 sigma confidence interval) of measured radiance." The 60-sec stability characterization requires scenes of temporally uniform radiance above $L_{typical}$[9], e.g., long solar acquisitions. Long shutter data will also be analyzed to allow determination of the bias stability contribution to the overall stability. These datasets (L0r) at most will only be bias and non-linearity corrected. Additionally, solar data products (L1r) will allow characterization of the L1r product 60-second stability. In the processing flow, these collects should be regarded as "special" collects vs. "standard" dark or solar collects identifying the data collects types that will meet the scene(s) minimum length criteria required for this processing.

There are two options for the analyst for processing these Scenes
Option 1 – that is described in this ADD is the IAS processing that relies on Histogram statistics Characterization to collect the basic statistics on the Scenes and then populate the minimal radiometric stability metric as a standalone algorithm that is gathering information only on the 60sec stability.
Option 2 – Is the a Toolkit algorithm that operates on the datasets directly and computes the basic statistics from the image data and then populate an extended radiometric stability metric that is gathering information to any stability period up to 60sec (as selected by the analyst) The toolkit version of this stability characterization has a separate ADD and data files for validation.

The output produced by this option 1 algorithm depends on the input data provided, i.e. solar or shutter. Response and gain differences statistics across time intervals of 60 seconds are calculated from these uniform radiance scenes and stored in the characterization database. As reflected in the OLI Ops-Con special 60 seconds long solar acquisitions and related 60 seconds long dark collects will be made at the nominal integration time and their statistical characteristics calculated by Histogram Statistics Characterization at multiple level of processing will be used as input to this algorithm. This short-term stability characterization will produce five categories of outputs:

1) Dark Shutter Raw data band level change in stability
2) Solar bias and non-linearity corrected data band average stability
3) Solar product band average stability in radiance
4) Band average gain stability or L1R Product stability in %
5) Detector by detector level relative gain stability in %

The individual detector relative gain short-term stability will be used in the analysis for the non-uniformity requirement stability and track actual performance characteristics against the system engineering allocations.
Solar L1r data products histograms will be used to directly verify the 0.5% radiometric stability requirement. The solar L1r product stability metric will be converted from radiance units to % so it

[9] These will be limited to Long Solar diffuser and Long Shutter collects only.

can be evaluated directly against the 0.5% requirement. Solar data that is bias and nonlinearity corrected will be used to characterize the 60s band mean and relative gains stabilities.

Note that the bias used for bias removal is derived from a model that is based on the pre and post solar collect event dark shutter collects which are also taken at the nominal integration time.
An additional Toolkit implementation for this characterization will exist. In that Toolkit implementation the algorithm will works directly on the image data and produce the basic Statistical information on the image or on a set of image subsets. It will generate more radiometric stability output categories for the investigating analyst.

7.5.9.2 Inputs

Descriptions	Symbol	Units	Level	Source
Signal Mean	$\overline{Q}_{B,S,D}$	Float [DN] or [w/m2 sr um]	N_BxN_SxN_D	Histogram[10] Stat Char $\overline{Q}$
Signal Max	$Max\_Q_{(B,S,D)}$	Float [DN] or [w/m2 sr um]	N_BxN_SxN_D	Histogram[10] Stat Char Q_{max}
Signal Min	$Min\_Q_{(B,S,D)}$	Float [DN] or [w/m2 sr um]	N_BxN_SxN_D	Histogram[10] Stat Char Q_{min}
Signal StDev	$Sigma\_Q_{(B,S,D)}$	Float [DN] or [w/m2 sr um]	N_BxN_SxN_D	Histogram[10] Stat Char σ
Relative gains	r	[unitless]	N_BxN_SxN_D	CPF
Scene Type (i.e O* or D*)	Stype	Char	N_B	DB L0Rp Image file
Processing Step (i.e. before or after Gain application)	Pstep	Integer	N_B	Histogram Stat Char Position in processing flow (RPS Level)
Impulse Noise Pixels Locations	LM1	Integer	N_BxN_SxN_D xL	LM
Saturated Pixel Mask[11]	LM2	Integer	N_BxN_SxN_D	LM

[10] Histogram data will be collected from different level of processing i.e. L0R for Dark Shutter data, L0Rc for Solar Diffuser and at L1R for Solar Diffuser product.
[11] This mask should include any of the detectors that had been reported by the Saturation characteristics processing, i.e. high and low saturations in both digital and analog categories

Dropped Frames Mask	LM3	Integer	N_BxN_SxN_D xL	LM
Inoperable Detector List	Dinop	Float	N_BxN_S	CPF

7.5.9.3 Outputs

Descriptions	Symbol	Units	Level	Target
Signal Variability, SCA average	$\overline{\Delta Q_{B,S}}$	Float [DN] or [w/m2 sr um]	N_B x N_S	DB (Bias, Gains & Solar L1R Stability)
Gain or L1r Product Variability, SCA average	$\overline{\Delta_{B,S}}$	Float [%]	N_B x N_S	DB (Gains & Solar L1R Stability)
Relative gain or L1r Product Variability, per-detector [12]	$\overline{\Delta_{B,S,D}}$	Float [%]	N_B x N_SxN_D	DB (Gains & Solar L1R Stability)
Scene Type (i.e O* or D*)	Stype	Char	N_B	DB
Processing Step	Pstep	Integer	N_B	DB

*Note: The developer should consider splitting these outputs into 2 or more DB tables; one for bias stability and the other for Gain stabilities. It would keep the units for each table consistent.[13]

7.5.9.4 Options

Trending On/Off Switch: If trending is Off, output parameters are written to a text file.

7.5.9.5 Prototype Code

The source code is written in IDL

```
IDL Version 7.0.8, Mac OS X (darwin x86_64 m64). (c) 2009, ITT Visual Information Solutions
```

It was test on iMac OS-X 10.6.1 3.06 GHz Intel Core 2 Duo

[12] We will not store in DB the Solar L1R calculated per detector gains 60 stabilities but it will be calculated and written to an output file if the option of trending is turned off.

[13] Note that the new output table consolidated two solar processing outputs into a single variable that have the same dimension, while the meaning of the stored data depends on the input data. This way we store both the percent of SCA absolute gain change and the percent SCA mean radiance change into a single DB column and similarly we store the percent per detector relative gain change and radiance signal stabilities in another DB column (this is possible since both are sharing the same dimension and have nearly the same calculations).

Source code include three modules from which one module serves as the main program that uses the others.
The instructions how to run each test scenario are put as comments in the First page of the main module.

Data outputs and inputs to modules at times use temporary IDL .sav files to store calculated variables. – those that could be generated by the developer they are not include in the package.

For each data type the data storage of what would be the link to DB is save into a IDL .sav file formats– those output files were saved and will be attached in with the source code

For L0r Dark data the generated parameters are store in - > IAS_Dark _output
For L0r Solar data the generated parameters are store in - > IAS_Solar _output1
For L1r Solar data the generated parameters are store in - > IAS_Solar _output2

OUTPUT variables generated ARE not in the same data format as it should be for the fully developed system since it is only dealing with one SCA and one band therefore 2 dimensions are missing.

The list of source code modules:

IAS_60SRS_main.pro - > main module
Get_stat.pro -> main stat calculation and ingesting of Histogram data module
IAS_cal_stab_solar.pro -> calculate the Solar related output stability performance parameters

Data inputs: dark_shutter_hist , solar1_hist , solar2_hist , rel_gains.sav

Dark_shutter_hist is the histogram data from dark shutter data (it holds information on Mean, Min, Max, and Stdev)

Solar1_hist is the histogram data from L0rc Bias removed and linearity correct L0r Solar data (it holds information on Mean, Min, Max, and Stdev)

Solar2_hist is the histogram data from L1r Solar Diffuser Product data (it holds information on Mean, Min, Max, and Stdev)

Data outputs:
Three folders: Test ID#1, Test ID#2, Test ID#3
Each holding the corresponding output: IAS_Dark_output, IAS_Solar_output1, IAS_Solar_output2 All stored in IDL .sav file formats

7.5.9.6 Procedure

Note that in order to characterize radiometric stability we call this algorithm at least 3 times: Once to process long dark shutter histogram data, and twice to process the histogram data for the closest long solar collect (once using the L0rc1 intermediate Cal product histogram results and once using the L1R intermediate Cal product histogram results.)

The trend ID's for long dark and long solar collects processing need to be found. Based on these trend ID's and the processing level the correct histograms information can be imported to this algorithm.

Calculating statistics (get it from Hist Stat)

---

1. Based on the Labeled Mask and detector operability list omit those detectors when calculating SCA level averages.
2. Populate histogram statistics metric per detector in the relevant variables and calculate the average across all detectors within an SCA.

   a. $\overline{Q}_{(B,S,D)} = Histogram\_per\_detector\_mean\_signal\,\overline{Q}$ (1)

   b. $Max\_\overline{Q}_{(B,S,D)} = Histogram\_per\_detector\_Max\_signal\ Q_{max}$ (2)

   c. $Min\_\overline{Q}_{(B,S,D)} = Histogram\_per\_detector\_Min\_signal\ Q_{min}$ (3)

   d. $Sigma\_\overline{Q}_{(B,S,D)} = Histogram\_per\_detector\_StDev\ \sigma$ (4)

   e. $\overline{Q}_{B,S} = mean(\overline{Q}_{B,S,D})$ (5)

   f. Note that calculations should only include operable and in-spec detectors. (i.e. ignore pixels flagged as inoperable, saturated, dropped frame, impulse or fill.)

Calculating Stability of Signal

---

3. For each detector, calculate the 2-sigma the along track variation in the image that is at the length of Δt=60sec.

   a. $\Delta Q_{(B,S,D)} = 2 \times Sigma\_\overline{Q}_{(B,S,D)}$ (6)

Processing data to generate outputs

---

4. Calculate the SCA average variability, i.e, the average across all detectors within each SCA:

   a. $\overline{\Delta Q_{B,S}} = mean(\overline{\Delta Q_{B,S,D}})$ (7)

5. For dark data, record the SCA mean variabilities ($\overline{\Delta Q_{B,S}}$) to the database or output file (along with other specified outputs in the output table). This is the end of the algorithm for dark data.
6. For solar data also do this :

   a. Calculate the per-detector variability in terms of percent change. It illustrates the impact of individual detectors on the overall radiometric stability.

      i. $\overline{\Delta_{B,S,D}} = r_D * \frac{\overline{\Delta Q_{B,S,D}}}{\overline{Q}_{B,S,D}} * 100\%$ (8)

ii. where r_D is the relative gain from CPF for L0Rc data and 1.0 for L1R data.

b. Calculate the SCA average variability in terms of percent gain change:

i. $$\overline{\Delta_{B,S}} = \frac{\overline{\Delta Q_{B,S}}}{\overline{Q}_{B,S}} * 100\% \quad (9)$$

ii. $\overline{Q}_{B,S}$ should be calculated using the same list of detectors used to produce the mean signal in equation (5)

c. For L0Rc data, write the per-detector percent variability ($\overline{\Delta_{B,S,D}}$) and the SCA average variability ($\overline{\Delta_{B,S}}$) to the database or output file (along with other specified outputs in the output table).

d. For L1R data, write the SCA average percent variability ($\overline{\Delta_{B,S}}$) to the database or output file (along with other specified outputs in the output table).

e. For L1R if output to file selected also write to file the per-detector variability ($\overline{\Delta_{B,S,D}}$) and the SCA Radiance mean variability ($\overline{\Delta Q_{B,S}}$) (along with other specified outputs in the output table).

7.5.9.7 Maturity

Level 2 (no reuse).

Algorithm design is settle on the fact that we will have especially long solar and dark collects that are at least 60sec in length.

Equation 6 was replaced to reflect BATC 2-sigma method to asses 60-sec stability that is based on the calculation of 60sec interval Standard Deviations statistics σ per detector over the entire interval of Δt=60sec. For that reason we elected to break this characterization into 2 flavors where the 1st is IAS standalone implementation where the statistics needed for this algorithm is calculated and passed directly from the relevant Histogram statistics Characterization DB, and the 2rd flavor is the Toolkit implementation of the algorithm where calculations of the basic statistics run directly on the data and more complicated options of trending and processing are implemented. The changes implemented for the IAS version is a simplification of original calculations.

Future direction of the algorithm is dependent on OLI performance and an over time comparison between the basic 2-sigma based calculation stored in the IAS and the Toolkit Worst-case based calculations. It could be that after the IOC period we will find a credible need for update of the IAS algorithm implementation, in that case we will submit a CCR to this and maybe histogram statistic ADDs.

7.5.10 OLI Detector Response Characterization (Lamp)

7.5.10.1 Background

The OLI will have 2 stimulation lamp fixtures. Each fixture will contain 6 tungsten lamps. A lamp in each fixture is paired (wired in series with) a lamp in the other fixture. Of the 6 lamp pairs, 3 are designated primary (operate on side A electronics) and 3 redundant (operate on side B electronics). Only one lamp pair is to be powered at a time. One pair of lamps on each side of the electronics with be designated "WORKING" with data to be acquired on a daily basis once operational. One pair of lamps will be designated "BACKUP" with nominally monthly usage and one pair "PRISTINE" with

semi-annual usage . Use of any redundant lamp can only occur when the full instrument is switched to the B-side electronics.

The current OLI concept of operations stipulates that the on-board lamps will be exercised daily. Data acquisition at the typical line rate of 4.236 ms/frame will begin several seconds prior to lamps turn-on, proceed with 2.75 minutes of lamp data, and continue until several seconds after the lamps have been turned off, yielding a total of approximately 3.34 minutes of data. However, the intent of the present algorithm is to provide trending statistics for characterizing post lamp warm-up detector responses. Therefore this algorithm will operate on 512(default value) lines of data taken 2.5 minutes after lamps turn-on and populate the database with the following statistics:

a) the mean, standard deviations, skewness and kurtosis at the detector level,
b) the averaged mean and standard deviation per band per sensor chip array (SCA) level, and
c) the averaged mean and standard deviation per band

The full 2.75 minutes of daily lamp data may be used for further lamp stability studies, or for assessment of Focal Plane Array (FPA), Focal Plane Electronics (FPE) or Lamp sub-system performances . Multiple within-orbit lamp collects separated by off time may also be prescribed for special study. Analysis of these extended datasets and lamp collects will be conducted off line as they are beyond the scope of the present algorithm. Associated telemetry from the stimulation lamps may be required to support the detector characterizations, for example:
Lamp Housing Temperatures (2), Monitoring diode outputs (2), Lamp currents, Lamp voltages, monitoring diode temperatures (2), selected lamp, lamp on-time. These telemetry data points will be included in the ancillary data within the wideband data.

The initial effective lamp radiance for each detector (to be provided by BATC) will be available in the CPF. Using the appropriate lamp telemetry parameters, these will be corrected for the on-board conditions during the lamp imaging event and the result stored in the database for comparison with the Level1R statistics.

7.5.10.2 Input

Descriptions	Symbol	Units	Level	Source	Type
Scene/Interval ID		unitless	1	dB (ancillary or MOC report)	Text
Bias and non-linear corrected (L0rc)		DN	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Integer
Lamp Level1		Scaled DN or radiance units	N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}		Integer or Float
Impulse Noise			N_{bands} x N_{SCAs} x $N_{detectors}$	LM	Integer
Saturated Pixels			N_{bands} x N_{SCAs} x $N_{detectors}$	LM	Integer
Inoperable Detectors			N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Integer
Dropped Frames			N_{bands} x N_{SCAs} x $N_{detectors}$ x N_{frames}	LM	Integer
Selected Lamp Pair		unitless	1	dB (ancillary)	Integer
Data Start Time		UTC	1	dB (ancillary or MOC report)	*
Lamp on/off time		UTC	1	dB (ancillary or MOC report)	*
Start and number of lines to process Statistics start time (Time offset from lamp turn on)		Frame number	2	ODL or dB(ancillary or MOC report)	Integer
Statistics data block size (nominally 512)			1	ODL or dB	Integer
Data Frame rate		Samples	1	dB	Float

		per second		(ancillary or MOC report)	
Lamp current		Amps	$N_{telemetry}$	dB (ancillary or MOC report)	Float
Lamp voltage		Volts	$N_{telemetry}$	dB (ancillary or MOC report)	Float
Monitor voltage		Volts	$N_{telemetry}$	dB (ancillary or MOC report)	Float
Lamp Housing Temperature		degrees Celsius	$N_{telemetry}$	dB (ancillary or MOC report)	Float
Monitor Diode temperatures		degrees Celsius	$N_{telemetry}$	dB (ancillary or MOC report)	Float
Focal Plane Temperatures		Kelvins	2	dB (ancillary or MOC report)	Float

* UTC in hhmmss.sss

7.5.10.3 Output

Descriptions	Symbol	Units	Level	Target	Type
Scene/Interval ID		unitless	1	dB	Text
Selected Lamp Pair		unitless	1	dB	Integer
Lamp on Time		Seconds	1	dB	Float
Processing interval:					
Start/Stop Time		UTC		dB	*
Start/Stop Frame/Line Numbers		unitless		dB	Integer
Lamp Level0c statistics (mean, stddev, skewness, kurtosis): a. Per Band per SCA per Detector		DN	N_{bands} x N_{SCAs} x a)$N_{detectors}$ b) N_{bands} x N_{SCAs}	dB	Float

b. Per Band per SCA c. Per Band			c) N_{bands}		
Lamp Level1R statistics (mean, stddev, skewness, kurtosis): a. Per Band per SCA per Detector b. Per Band per SCA c. Per Band		DN	a)N_{bands} x N_{SCAs} x $N_{detectors}$ b) N_{bands} x N_{SCAs} c) N_{bands}	dB	Float
Lamp Current A:					
Mean		Amps	1	dB	Float
Sigma		Amps	1	dB	Float
Lamp Voltage A:					
Mean		Volts	1	dB	Float
Sigma		Volts	1	dB	Float
Monitor Voltage A:					
Mean		Volts	1	dB	Float
Sigma		Volts	1	dB	Float
Lamp Housing Temperature A:					
Mean		Degrees Celsius	1	dB	Float
Sigma		Degrees Celsius	1	dB	Float
Monitor Diode Temperature A:					
Mean		Degrees Celsius	1	dB	Float
Sigma		Degrees Celsius	1	dB	Float
Lamp Current B:					
Mean		Amps	1	dB	Float
Sigma		Amps	1	dB	Float
Lamp Voltage B:					
Mean		Volts	1	dB	Float
Sigma		Volts	1	dB	Float
Monitor Voltage B:					
Mean		Volts	1	dB	Float
Sigma		Volts	1	dB	Float
Lamp Housing Temperature B:					
Mean		Degrees	1	dB	Float

		Celsius			
Sigma		Degrees Celsius	1	dB	Float
Monitor Diode Temperature B:					
Mean		Degrees Celsius	1	dB	Float
Sigma		Degrees Celsius	1	dB	Float
Selected Lamp			1	dB	Integer
Focal Plane Temperatures at FPM7 and FPM14:				dB	Float
FPM 7 Mean		Kelvin	2	dB	Float
FPM 7 Sigma		Kelvin	2	dB	Float
FPM 14 Mean		Kelvin	2	dB	Float
FPM 14 Sigma		Kelvin	2	dB	Float

* UTC in hhmmss.sss

7.5.10.4 Options

Lamp warm-up interval input – Specifies Time (UTC) and image line number interval allowed for lamp warm-up.

Report outputs – Enables output of statistics results and telemetry data (eg. Temperatures, diode voltages) in text format.

7.5.10.5 Procedure

I. Read L0c and Level1R data.

a. Calculate frame number at 2.5 minutes (stats_starttime) from lamp turn on, and frame number at end of statistics data block (default stats_nsamples=512):

i. Stats_startframe=framerate*stats_starttime

ii. Stats_endframe=stats_startframe+stats_nsamples

b. Extract data block from stats_startframe to stats_endframe from the appropriate data files.

II. For L0c and Level1R lamp statistics data block do:

a. Calculate statistics for each detector (det), band, SCA.

i. Initialize statistics arrays:

$$Mean(\bar{x}), S\tan dard deviation(\sigma), Skewness(S), Kurtosis(k) = float(n\det\_ms\ or\ n\det\_pn, nBands, nSCA),\ where\ nBands = 9,$$
$$nSCA = 14, n\det, n\det_{ms} = 494,\ n\det_{pn} = 988$$

ms denote ms bands, *pn* denotes pan band

ii. For each detector, Band, SCA compute:

1. Mean, $\bar{x}_{(\det,band,SCA)} = \frac{1}{nFrames}\sum_{i=StartFrame}^{EndFrame} x_i$,
   $where\ nFrames = EndFrame - StartFrame$
2. Standard Deviation, $\sigma_{(\det,band,SCA)} = \sqrt{\frac{1}{nFrames}\sum_{i=StartFrame}^{EndFrame}(x_i - \bar{x})^2}$
3. Skewness, $S_{(\det,band,SCA)} = \frac{\left(\frac{1}{nFrames}\sum_{i=StartFrame}^{EndFrame}(x_i - \bar{x})^3\right)}{\sigma^3}$
4. Kurtosis, $K_{(\det,band,SCA)} = \frac{\left(\frac{1}{nFrames}\sum_{i=StartFrame}^{EndFrame}(x_i - \bar{x})^4\right)}{\sigma^4} - 3$

b. Calculate band averages per SCA means ($\bar{X}$) and standard deviations($\bar{\sigma}$):

i. Initialize data arrays, $\bar{X}, \bar{\sigma} = float(nBands, nSCA)$

ii. For each Band, SCA compute:

1. $\bar{X}_{(Band,SCA)} = \frac{1}{n\det}\sum_{\det=0}^{n\det-1}\bar{x}_{(\det,band,SCA)}$
2. $\bar{\sigma}_{(Band,SCA)} = \frac{1}{n\det}\sum_{\det=0}^{n\det-1}\sigma_{(\det,band,SCA)}$

c. Calculate per band mean and standard deviation. .

i. Initialize data arrays, $LampMean = float(nBands)$, $LampStdev = float(nBands)$

ii. $LampMean_{(Band)} = \frac{1}{nSCA}\sum_{fov=0}^{nSCA-1}\bar{X}_{(band,SCA)}$

iii. $LampStdev_{(Band)} = \frac{1}{nSCA}\sum_{fov=0}^{nSCA-1}\bar{\sigma}_{(band,SCA)}$

III. Populate database with the L0c and Level1R statistics..

IV. Populate database with selected lamp telemetry

a. Lamp voltages

b. Lamp currents

c. Diode currents

d. Lamp housing temperatures

7.5.10.6 Maturity

Level 2 (no heritage) Changes may be required once on orbit.

Include detector responses at 30s intervals during the lamp warm-up period.

7.5.11 OLI Lunar Irradiance Characterization

7.5.11.1 Background

Lunar acquisitions will be used to complement image quality assessments of the OLI. These will help detect changes in gain, provide a measure of radiometric stability, and reduce absolute radiometric uncertainties.

Changes in relative gains are determined by comparing the measured lunar irradiances with modeled irradiances, which are calculated by the Robotics Lunar Observatory (RoLO), USGS/Flagstaff, using their lunar irradiance model. The interface between IAS and RoLO consists of a set of data interchange files. The IAS provides RoLO with the measured integrated lunar irradiances, image times and spacecraft position vectors. RoLO in turn generates a set of reports containing lunar observation geometrical parameters used by the model and comparisons of suitably scaled measured versus modeled irradiances. These reports will be ingested into the database for trending.

The L1R lunar product will also be used for further geometrical processing and analysis including creation of geometrically corrected image products (resampled) and MTF characterizations.

Operationally, it is expected that the moon will be imaged once a month (at approximate 7 degrees lunar phase angle) on several Sensor Chip Arrays (SCAs) with one “reference” SCA image always acquired. Information regarding which SCAs are illuminated is also expected from mission operations element since they will be programming the lunar maneuver.

Input

Descriptions	Symbol	Units	Level	Source	Type
Scene (L1r)		$W/m^2/sr/\mu m$	N_{bands} x N_{SCAs} x $N_{detectors}$		Float
ImpNoise			N_{bands} x N_{SCAs} x $N_{detector}$	LM	Int

			s		
Sat Pixels			N_{bands} x N_{SCAs} x $N_{detectors}$	LM	Int
Inop Dets			N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	Int
Odd/even detector offsets			N_{bands} x N_{SCAs} x $N_{detectors}$	CPF	
Median Filter Size	β			CPF	Int
Dropped Frames			N_{bands} x N_{SCAs}	LM	Int
SCA_illumination_flag			N_{SCAs}	Currently not available in the Ancillary/db. A mechanism to get this information is desired.	Int
Data/Imagery start/stop time [UTC]	T_{start},T_{end}	YYYYDDDhhmmss.sss		Ancillary/db	Text or Date
Frame rate	F	Hz		Ancillary/db	Float
Integration time	I	Ms		Ancillary/db	Float
Spacecraft ephemeris/Attitude		Km		Ancillary/db	Float
Moon-SC distance	R_{m-sc}	Km		Currently not available in Ancillary/db but can be calculated by IAS using existing code.	Float
Earth-Sun distance	R_{e-s}	Km		Currently not	Float

				available in Ancillary/db but can be calculated by IAS using existing code.	
Earth-Moon distance	R_{e-m}	Km		Currently not available in Ancillary/db but can be calculated by IAS using existing code.	Float
Earth-SC distance	R_{e-sc}	Km		Currently not available in Ancillary/db but can be calculated by IAS using existing code.	Float
Radiance Integration Thresholds	δ		N_{bands}	CPF	Float

Output

Descriptions	**Symbol**	**Units**	**Level**	**Target**	**Type**
Interface items to ROLO					
Time at center of lunar image [UTC]	t_m	YYYYDDDhhmmss.sss	N_{bands} x N_{SCAs}	DB and text file	Text or Date
Apparent lunar YSize [mrad]	Ysize	mrad	N_{bands} x N_{SCAs}	Db and text file	Float
Integrated Irradiances	SumIrr	[microWatts / m^2 / nm]	N_{bands} x N_{SCAs}	Db and text file	Float
SC PositionVector [J2000 GCI]		Km		Db and text file	Float

Interface items from ROLO					
Modeled Integrated Irradiances		microW/ m^2/nm	N_{bands} x N_{SCAs}	Db and text file	Float
Scaled Measured Irradiances		microW/ m^2/nm	N_{bands} x N_{SCAs}	Db and text file	Float
Irradiance percent differences		percent	N_{bands} x N_{SCAs}	Db and text file	Float
Lunar diameter as observed from the SC		mrad	N_{bands} x N_{SCAs}	Db and text file	Float
Oversampling Factor		dimensionless	N_{bands} x N_{SCAs}	Db and text file	Float
Dynamical barycentric Days		Day		Db and text file	Float
Selenographic longitude of the Sun		degrees		Db and text file	Float
Selenographic latitude of the Sun		degrees		Db and text file	Float
Selenographic longitude of spacecraft		degrees		Db and text file	Float
Selenographic latitude of spacecraft		degrees		Db and text file	Float
Distance of spacecraft from center of Moon		km		Db and text file	Float
Heliocentric range of the Moon		Au		Db and text file	Float
Factor to correct irradiance to standard distances		dimensionless		Db and text file	Float
Lunar phase angle		mrad		Db and text file	Float
Position Angle of lunar axis, ccw from North		degrees		Db and text file	Float
Lunar Mask		boolean	N_{bands} x N_{SCAs} x $N_{detectors}$	Image File	Int
Other algorithm generated parameters					
Frame numbers at top and bottom of lunar image, location moon's center (frame number, detector location)		N/A	N_{bands} x N_{SCAs}	DB and text file	Int

S/C attitude and attitude rates at top, middle and bottom of lunar image		rads,rads/s or arcsec,arcsec/s [as provided in the ancillary db]	N_{bands} x N_{SCAs}	DB and text file	Float
Sun-Earth-Moon ranges [km]		km		DB and text file	Float

Note, Ints are 16-bit, Floats are 32-bit.

7.5.11.2 Options

1. Output lunar mask for verification.
2. Text output of Integrated Irradiance, apparent lunar y-sizes, start and stop frame numbers of lunar images, and location of the middle of the lunar images for each band and in every illuminated SCA.
3. Alternate median filter size.

7.5.11.3 Procedure

I. Obtain illuminated SCA flag from ancillary database or other methods/sources.

II. For each illuminated SCA calculate inputs to Rolo

A. Read L1R metadata
   i. Image dimensions
   ii. Image start time
   iii. Frame rate
   iv. Integration time

B. For each BAND
   i. Read image data, $L1R_{iband,isca}$
   ii. If the input L1R data implemented any odd even offsets, then remove odd/even detector offsets from $L1R_{iband,isca} \rightarrow L1C_{iband,isca}$. Otherwise, $L1C_{iband,isca} = L1R_{iband,isca}$
   iii. Create Lunar Mask, $LM_{iband,isca}$:
      By thresholding.
      1. Remove artifacts (bright stars, etc) from image, eg: Filter each column with median filter, size = β. (default=5).
      2. Obtain maximum radiance value, $MaxRad_{iband,isca} = max(L1C_{iband,isca})$
      3. Set irradiance Threshold = $MaxRad_{iband,isca} * \delta$, where δ is the radiance integration threshold factor (current default value = 0.8)
      4. LM = 1 where $L1C_{iband,isca} \geq$ Threshold, else LM = 0
   iv. Calculate Irradiance, $SumIrr_{iband,isca}$
      1. Sum radiance over lunar images, $SumRad_{iband,isca} = \sum LRad*LM$.
      2. Convert $SumRAD_{iband,isca}$ to Irradiance in $\mu W/(m^2.nm)$, SumIrr:
         For MS bands, $SumIrr_{(1-7,9),isca}=SumRAD_{(1-7,9),isca}$* 1.81077e-13
         For Pan band, $SummIrr_{8,isca}=SumRAD_{8,isca}$*4.52694e-14

These constant may be added to the CPF.

v. Calculate apparent lunar y-size:
   1. For each detector locate start/end frame number where LM = 1, $start_{idet}$, end_{idet}
   2. Number of mask pixels in each column: $N_{idet}=end_{idet}-start_{idet}$
   3. $Ysize_{iband,isca}$ = Maximum(N_{idet}), at column location $Dmax_{iband,isca}$. .I.e. Y-size is the maximum number of pixels set to 1 in the lunar mask and Dmax is the detector number where that maximum occurs.
   4. Ysize in mrads,:
      For MS bands, $mYsize_{(1-7,9),isca} = Ysize_{(1-7,9),isca} * \Omega_{ms}$
      where detector solid angle is Ω_{ms}=4.2553e-5
      For Pan band, $mYsize_{8,isca} = Ysize_{8,isca}*\Omega_{pan}$
      where detector solid angle is Ω_{pan}=2.1277e-5

vi. Calculate UTC time at center of moon, UTC_Moon(Band,SCA):
   1. Center of moon in image, FrameNumberMoonCenter = (start(Dmax)+end(Dmax)/)/2
   2. UTC_moon=(Image start time)+FrameNumberMoonCenter/(Frame rate)
   3. Alternatively, query image data time code at FrameNumberMoonCenter.

vii. Read spacecraft ephemeris at UTC_moon: J2000 Position Vector(X,Y,Z).

III. Outputs

Obtain and populate database with integrated irradiance results and ancillary data. These include:

i. Sun-moon-earth ranges,
ii. Lunar Irradiance,
iii. Lunar Y-sizes (in mrads and frame count),
iv. Frame numbers at top and bottom of lunar edges.
v. UTC time, spacecraft position vectors at center of lunar image.

IV. Rolo interface:

i. Query database, format and transmit results to Rolo.
ii. Receive lunar model results from Rolo and populate database.

Known/Potential Issues:

One input specification is a flag to denote which SCA is being illuminated. Currently, such information is not available to IAS. An alternate, but less preferred, method is for IAS to use the imagery itself to determine which SCA is illuminated.

The Moon-SC, Earth-Sun, Earth-Moon, and Earth-SC distances are also not currently available in the ancillary databases. The IAS can alternatively calculate this information using existing software.

7.5.11.4 Maturity

1. Level 2 (reuse from ALIAS)
2. Enable usage of L1G as input.
3. Develop alternate methods for obtaining lunar masks. Eg
    a. Image classification to separate moon image from dark space.
    b. Lunar edge/limb determination as implemented for the RoLO model.
4. Estimate apparent lunar y-size using spacecraft attitude information. This requires spacecraft attitude and attitude rates(Roll, Pitch, Yaw), and moon size from the spacecraft's position.

7.5.12 OLI Reflectance Conversion

7.5.12.1 Background

The standard Level 1T product will be a top of atmosphere reflectance product. This algorithm will convert the radiance image to a reflectance image in a per-scene operation. The two products are linearly related to each other by a band specific coefficient that is proportional to the exoatmospheric solar irradiance in each band and the Earth-Sun distance for the scene's day of acquisition. The per-band coefficients will be determined once on orbit, after the first look at the diffuser. For prelaunch testing, an estimate of the coefficient can be derived from the exoatmospheric solar irradiance. The reflectance values will be between 0.0 and 1.0.

Since all problem pixels should have been corrected by this point in the processing flow, this algorithm assumes that every image pixel is a valid radiance value. Thus there is no consideration for dropped frames, inoperable detectors or saturated pixels.

This algorithm will only process OLI data, not TIRS data. The equivalent algorithm for TIRS data is Temperature Conversion.

7.5.12.2 Inputs

The inputs to this algorithm are the image, parameters from the CPF and a parameter from the JPL ephemeris table. Table 12 lists the inputs of this algorithm.

Table 16: Algorithm Inputs

Descriptions	Symbol	Units	Level	Source	Type
Radiance image data	L	W/m^2 sr um	N_{band} x N_{SCA} x N_{det} x N_{frames}		float
Earth-Sun Distance	d	AU	scalar	JPL ephemeris table	float
Radiance to Reflectance Conversion Coefficients	Rr	sr/(w/m^2 um)	N_{band}	CPF	float

7.5.12.3 Output

The outputs of this algorithm are the reflectance image and the coefficients needed to convert back to radiance. The coefficients will be distributed with the final product. Table 13 lists the outputs of this algorithm.

Table 17: Algorithm Outputs

Descriptions	Symbol	Units	Level	Target	Type
Reflectance image	P	[]	N_{band} x N_{SCA} x N_{det} x N_{frames}		float
Reflectance to radiance conversion coefficients	ρ_R	(w/m^2 um)/sr	N_{band}	Radiance Rescaling	float

7.5.12.4 Options

7.5.12.5 Procedure

1. For each band, apply conversion coefficients (Rρ) to radiance images (L1r). This equation converts the image radiance [W/m^2 sr um] to reflectance [unitless].

   a. $\rho = d^2 \cdot L \cdot R_\rho$ (1)

   b. where L is the image radiance and d is the earth-sun distance (AU) for the day the scene was acquired.

2. For each band, calculate reflectance-to-radiance conversion coefficient (ρ_R).

   a. $\rho_R = \frac{1}{d^2 \cdot R_\rho}$ (2)

   b. This coefficient is passed to Radiance Rescaling.

7.5.12.6 Inputs

Radiance-to-reflectance coeff:	$R\rho = 0.00167$	(arbitrary value)
Radiance image mean:	L = 319.2	(an arbitrary high signal)
Earth-sun distance:	d = 1.0154351	(distance for June 12, an arbitrary date)

7.5.12.7 Outputs

Reflectance image mean:	ρ = 0.550
Reflectance-to-radiance coeff:	ρ_R = 580.737

7.5.12.8 Maturity

Level 2 . It is unlikely but another version of this algorithm may need to be implemented during the level 1G processing (or after), if the decision is made to generate a reflectance product where the sun angle used is specific to the image pixel. Here, a latitude and longitude of each pixel combined with the image time, will be used to calculate a reflectance for the specific sun angle corresponding to each pixel.

7.5.12.8.1 Earth-Sun Distance Calculation

The JPL Ephemeris table (DE421) describes the orbits of the sun and planets with very high precision over relatively long time scales[14]. The file is stored as a series of Chebyshev coefficients which can be interpolated to essentially any desired temporal accuracy. The IDL tool that I use to read from the JPL file requires the date as input and outputs a three element position array. To convert to earth-sun distance in AU:

```
aukm = double(1.4959787e8) ; au in km
dist = double( sqrt( pos(0)^2 + pos(1)^2 + pos(2)^2. ) )
earth_sun_distance = dist / aukm
```

At this point, it is unclear what form the input to this algorithm will be. Will it already be the earth-sun distance or will it be passed in as the 3-element Chebyshev coefficients? I'm still presuming the earth-sun distance will be passed in, but if not, these equations will need to be added as Step 1.

7.6 TIRS Radiometry Algorithms

7.6.1 TIRS Dark Response Determination

7.6.1.1 Background

One component of the signal recorded by each TIRS detector is the result of the response to the thermal energy in the QWIP material, called the Dark Response. This value varies by detector for all bands and is dependent on detector temperature, bias voltage, integration time and detector gain. Nominal values are determined during ground testing at the component level by using a cold shutter to mask practically all photon flux from the detectors. In-flight, the two Earth-imaging "illuminated" bands cannot be shuttered from the photons emitted by the instrument structure and optics, thus their dark response cannot be directly measured. A third TIRS band, the "dark" band, is permanently shuttered. The response for this dark band is continuously measured and available along with the illuminated band data. If there is variation in the dark response on orbit, e.g., as due to detector temperature variation, this dark band will capture it.

This algorithm analyzes the response from the dark band, compares it to the dark response from ground testing and calculates an estimate of the change in dark response. This estimate can be extrapolated to a change in the dark response of the illuminated band data and applied to the radiometric calibration in later steps. The application of the correction is dependent on the evaluation of how well the dark response of the dark band correlates to the dark response of the illuminated bands. Since the dark response of a QWIP detector is a function of its individual gain, the difference in gains between the dark and illuminated detectors must be accounted for in the evaluation.

This algorithm can be run on sets of calibration observations as a stand-alone algorithm to evaluate the dark response correlation and stability. It can also be run on a scene-by-scene basis as part of the primary radiometric calibration. In the former case, CPF values can be evaluated and adjusted to reflect the best in-flight knowledge of the dark response. This process will also be necessary if the set points for integration time, bias voltage, detector temperature, or A/D offset level are changed in flight. In the latter case, this algorithm is needed in-line for product generation.

The TIRS instrument is expected to be stable in terms of radiometric gain and bias such that calibration updates should not be required on an orbital or daily basis. In this scenario the dark response is treated as part of the overall background response and this algorithm may not substantially improve the precision of the radiometric calibration. In the event that the focal plane temperature shows low frequency variation, lower than per scene, this correction may improve the absolute radiometric calibration precision.

7.6.1.2 Input

Description	Symbol	Units	Level	Source	Type
Scene Mean (dark band only)	$\overline{Q}$	DN	N_{SCA} x N_{det}	DB (Histogram Statistics Characterization)	Float
Baseline Dark Response	D_B	DN	N_{band} x N_{SCA} x N_{det}	CPF	Float
Relative Gain Coefficients	G_R	Unitless	N_{band} x N_{SCA} x N_{det}	CPF	Float
Absolute Gain Coefficients	G_A	$\frac{DN}{W/m^2 sr \mu m}$	N_{band} x N_{SCA}	CPF	Float

7.6.1.3 Output

Description	Symbol	Units	Level	Target	Type
Dark Response	D	DN	N_{band} x N_{SCA} x N_{det}	Test CPF or Bias Removal	Float

7.6.1.4 Options

- Output adjusted dark response to file (default off). This is mainly for the standalone implementation of this algorithm, but could also be helpful for anomaly resolution.
- Output baseline dark response or adjusted dark response (default is baseline dark response from the CPF).

7.6.1.5 Procedure

5. Read the baseline dark response (D_B) for each band/SCA/detector from the CPF (including the dark band)
6. If baseline dark response option set, output the dark response (D_B) and exit, else continue
7. Read the gain coefficients (G_A and G_R) for each band/SCA/detector from the CPF (including the dark band) and calculate the per-detector absolute gain (G)

$$G(b,s,d) = G_A(b,s)G_R(b,s,d) \qquad (1)$$

8. Read the scene average response ($\overline{Q}$) for each SCA/detector for the dark band from the histogram statistics for the input scene or calibration collection being processed.
9. For each SCA:
   9.1. For each detector in the Dark band:
      9.1.1. Calculate the difference between the baseline dark response (D_B)and the scene average dark response ($\overline{Q}$), giving the dark response difference (Δ_D)

$$\Delta_D(s,d) = D_B(s,d) - \overline{Q}(s,d) \qquad (2)$$

      9.1.2. Divide the dark response difference (Δ_D) by the detector gain (G), giving the adjusted dark response difference (Δ_A)

$$\Delta_A(s,d) = \frac{\Delta_D(s,d)}{G(s,d)} \qquad (3)$$

      9.1.3. Calculate the average adjusted dark response difference (Δ_S)across all operable dark detectors in the SCA, $\Delta_S(s) = \frac{1}{N_d}\sum_{d=1}^{N_d}\Delta_A(s,d)$ (4)

   9.2. For each Band/Detector:
      9.2.1. Multiply the SCA average adjusted dark response difference by each detector's gain and add the result to the baseline dark response for each detector, giving the adjusted dark response for the detector.

$$D_A(b,s,d) = G(b,s,d)\Delta_S(s) + D_B(b,s,d) \qquad (5)$$

Downstream algorithms can use either the baseline dark response (D_B) or the adjusted dark response (D_A), in either case it is represented as simply the dark response (D).

7.6.1.6 Maturity

Level 3. Further component and higher level testing of the TIRS QWIP detectors is necessary before the details of this algorithm are resolved.

Depending on the stability of the dark response, we may need to store the results of the dark response determination in the characterization database.

There may also be a need to provide the output of this algorithm to the ICs in the form of a BPF.

7.6.2 TIRS Bias Model Calibration

7.6.2.1 Background

Conversion from instrument digital counts (DN) to radiance (W/m^^{2}-sr-µm) occurs in 3 steps: response linearization, bias removal, and gain application. The bias that is removed in the second step is a combination of the dark and background response of the instrument, and is the total response of the instrument to "nothing", or a very cold target. On orbit the TIRS instrument will collect data while looking at deep space. The per-detector means of the data from these collects can be used as an estimate of the cumulative dark and background responses of the instrument.
This algorithm only needs to be implemented as a part of Ingest, and should not be run using long (meaning longer than the typical 2 seconds) space look collects.

7.6.2.2 Input

Description	Symbol	Units	Level	Source	Type
Per Detector Means from latest space look (TIRS)	S_{TIRS}	DN	$N_{bands}xN_{SCA}xN_{detectors}$	Histogram Statistics	Float

utput

Description	Symbol	Units	Level	Target	Type
Per Detector Means from latest space look (TIRS)	S_{TIRS}	DN	$N_{bands}xN_{SCA}xN_{detectors}$	BPF	Float

ptions

- S

tart and stop date/time of desired means (T0 and T1). Normally T0 should be the stop of the pre-acquisition deep space collect andT1 should be the start of the post-acquisition deep space collect .

7.6.2.5 Procedure

Retrieve the histogram statistics from one collect each of TIRS space look data acquired prior to the desired start date/time (T0) and after the desired stop date/time (T1) and write these to the BPF database.

7.6.3 TIRS Bias Removal

7.6.3.1 Background

Conversion from instrument digital counts (DN) to radiance (W/m^^{2}-sr-µm) occurs in 3 steps: response linearization, bias removal, and gain application. The second step (bias removal) is described in this document. Although the name implies this algorithm just removes the detector bias, it actually removes the dark response and background response (which can come from the dark and background response determination algorithms or from per detector means of deep space look collects), as well as an offset that is part of the gain function that converts linearized, background subtracted DN to radiance. Note that the order of these three steps is different than OLI because the "bias" being removed is a function of detector voltage and temperature and must be linearized before being subtracted.

Bias removal is accomplished by subtracting a value (in linearized DN) from each pixel of the input image. This value varies by detector for all bands. The values are determined initially during ground testing and made available via the calibration parameter file (CPF). In-flight calibration observations

(space-look data) can be used to verify the per-detector background signal and adjust the CPF periodically if necessary.

7.6.3.2 Input

Description	Symbol	Units	Level	Source	Type
Scene (Linearized DN)	Q_L	DN	N_{band} x N_{SCA} x N_{det} x N_{frames}	Response Linearization	Float
Pre-acquisition Deep Space Averages	S_a	DN	N_{band} x N_{SCA} x N_{det}	BPF	Float
Post-acquisition Deep Space Averages	S_b	DN	N_{band} x N_{SCA} x N_{det}	BPF	Float
Dark Response	D	DN	N_{band} x N_{SCA} x N_{det}	Dark Response Determination or CPF	Float
Background Response	B	DN	N_{band} x N_{SCA} x N_{det}	CPF	Float
Gain Function Offset	Go	DN	N_{band} x N_{SCA} x N_{det}	CPF	Float

Output

Description	Symbol	Units	Level	Target	Type
Scene (Bias Subtracted, Linearized DN)	Q_{LB}	DN	N_{band} x N_{SCA} x N_{det} x N_{frames}	Gain Application	Float

7.6.3.3 Options

- Dark and Background Response Selection

1. Pre-acquisition deep space averages (S_a)
2. Post-acquisition deep space averages (S_b)
3. Average of pre-and post-acquisition deep space averages (S_{ab}) (default)
4. dark and background responses from the CPF (or dark response determination in the case of dark response)

7.6.3.4 Procedure

1. If pre- or post-acquisition deep space averages have been selected, then retrieve the selected deep space averages. Whichever is selected will be referred to as *S(b,s,d)* for the remainder of the algorithm, where *d*=detector, *s*=SCA, and *b*=band

   If the average of the pre-and post-acquisitions is selected then retrieve the selected deep space averages and calculate S using equation (1).

   $$S(b,s,d) = \frac{1}{2}\left(S_A(b,s,d) + S_B(b,s,d)\right) \quad (1)$$

   If the dark and background responses from the CPF are desired, then retrieve the dark and background responses from the CPF (or dark response determination in the case of dark response) and calculate *S* using equation (2).

   $$S(b,s,d) = D(b,s,d) + B(b,s,d) \quad (2)$$

2. To calculate the total bias, add the gain function offsets to the combined dark and background responses.

   $$bias(b,s,d) = S(b,s,d) + G_O(b,s,d) \quad (3)$$

3. Remove the bias from the linearized scene.

   $$Q_{LB}(b,s,d,f) = Q_L(b,s,d,f) - bias(b,s,d) \quad (4)$$

7.6.3.5 Maturity

This algorithm will likely be correct for the flight FPA. A native C implementation will be completed by the end of 2010 as part of the TIRS Science Data Processor pipeline, and the plan is to integrate it with the CalVal toolkit.

All of test data used was simulated or calculated from TVAC2 test data during a collect where the focal plane was looking at a cold plate meant to simulate deep space. The gain offsets, however were calculated from OBC data. The first useful in-flight Background Response CPF values will come from the “space-look” data collected during the 90 day checkout.

Expect to have more extensive test data sets as other algorithms are completed.

7.6.4 TIRS Nonlinear Response Characterization

7.6.5 TIRS Response Linearization

7.6.5.1 Background/Introduction

The TIRS readout electronics produce a quantized output (Q) in units of digital number (DN) representing the amount of signal measured by each detector. This Q is expected to be related to the input signal in a non-linear way. The non-linearity is different for each detector. To simplify parts of radiometric correction and calibration, the Q must first be adjusted to compensate for this non-linearity.

Response linearization data are collected using multiple integrations times with a stable signal source. During ground calibration the stable source is a shuttered dewar, a uniform calibration source, or the TIRS Calibration GSE. On-orbit, the stable signal source can be either the on-board calibrator or the space view. Plotting the Q vs. Integration Time illustrates the non-linearity in the detector response as the electron well fills. This methodology assumes that linearity with integration time is equivalent to linearity with radiance.

The 9803 Readout Multiplexers used in the TIRS focal plane have two distinct slopes. The initial response of output signal versus input signal has a relatively high slope for approximately the first quarter of the detector saturation level (low-end slope). The majority of the dynamic range of the detector/readout system (primary slope) has a response about 1/3 of the low-end slope. Nearing analog saturation the response slope flattens out (high-end slope). Additionally there are non-linearities in each response slope due to the ROIC output buffer electronics, and the analog to digital circuitry.

Figure 1 shows the response slopes for a typical detector/readout system from one of the TIRS SCAs. For convenience the Integration Time is normalized to the point where the primary slope output signal would be 4095 DN.

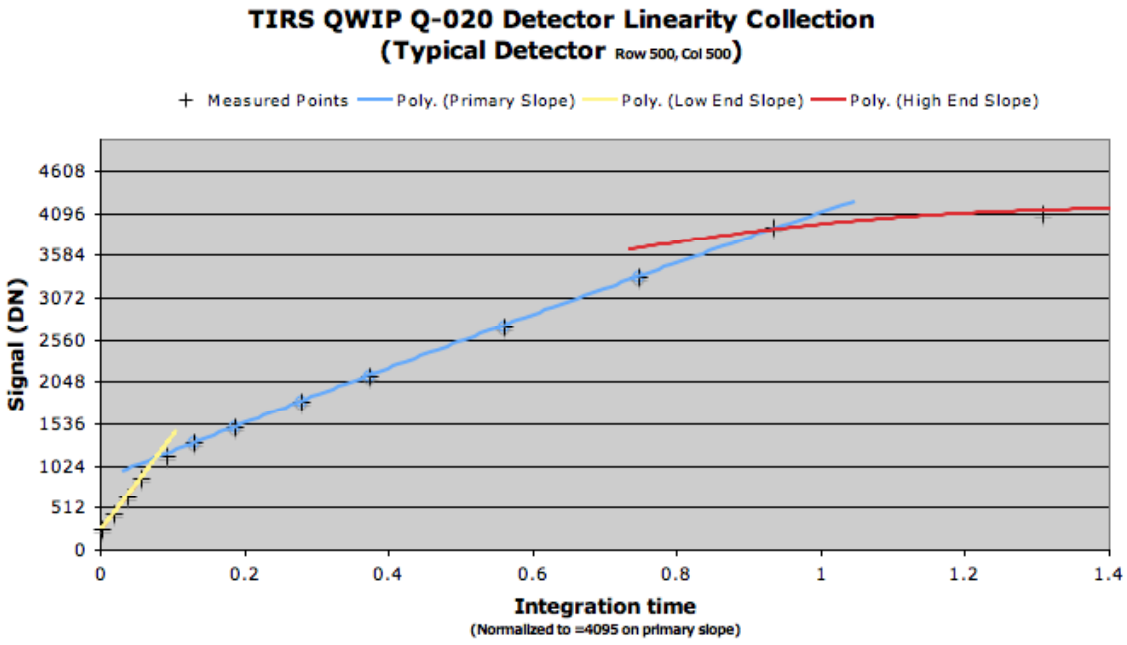
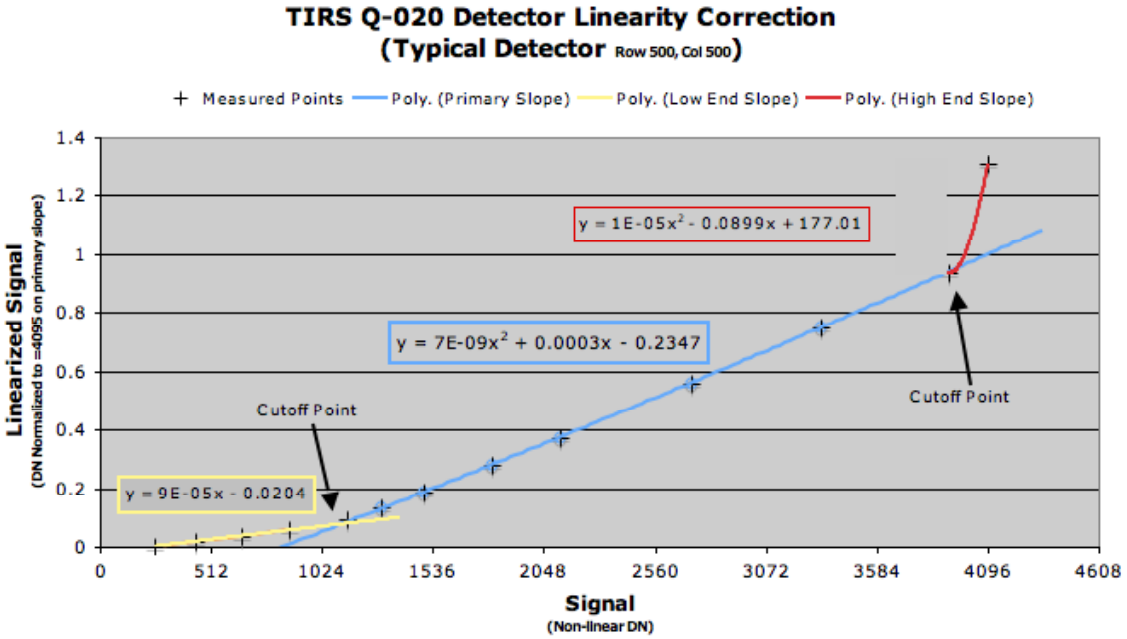


Figure 4. Typical QWIP Detector Response Shape

Figure 2 shows the inverted response plot with each response slope fit to a quadratic function. This data is derived from the DCL detector level data. There are not enough data points at this time to derive a quadratic function for the low-end slope, so the square term is set to zero. The high-end slope also lacks enough data points at this time for a good fit. The high-end slope is also likely to fit an exponential function better than a quadratic. The theoretical form of the curve is not known at this time, but the piecewise quadratic/linear approximation has been determined to be valid.



The instrument is designed so that the scene and OBC data will be in the primary slope portion of the response curve. Depending on the focal plane temperature, integration time, bias voltage, and other factors, the dark band data and deep space looks may return levels in the range of the low-end slope. In order to correct the scene data using the dark response it is necessary to match the response of the low-end slope to the primary slope. The cutoff point where the two slopes intersect is determined and the appropriate linearization correction is made. The instrument will likely be tuned so the analog saturation point associated with the high-end slope is above the digital saturation point, and thus these data will never be seen. Processing the high-end slope is included in case it is needed.

The mapping function used in this algorithm is a quadratic for all three slopes.. The cutoff points are determined by equating the two adjacent quadratics functions. Analysis has shown that a higher order polynomial does not add any benefit to the linearization to the low end slope or primary slope of the response curve

7.6.5.2 Input

Description	Level	Source	Type
Scene (L0R)	N_{scas} x N_{bands} x $N_{detectors}$ x N_{lines}		Float
Remapping function cutoff	N_{scas} x	CPF	Float

thresholds	N_{bands} x $N_{detectors}$ x $N_{cutoffs}$		
Remapping function parameters	N_{scas} x N_{bands} x $N_{detectors}$ x $N_{coefficients}$ x ($N_{cutoffs}$+1)	CPF	Float

The planned implementation is for two cutoff points ($N_{cutoffs}$ = 2) and three sets of quadratic coefficients ($N_{coefficients}$ = 3).

7.6.5.3 Output

Description	Level	Target	Type
Scene (linearized)	$N_{detectors}$ x N_{frames}		Float

7.6.5.4 Procedure

Response linearization should be a simple replacement of the incoming value with a linearized response value.

The linearized value will be calculated for each sample point by an algorithm. These algorithms will be determined by pre-launch calibration of the detectors. As described above, this algorithm is a multi-segment quadratic function with the detector Q value as its only variable. Parameters for the linearization function will be stored in the CPF. For this method the required parameters are three sets of quadratic coefficients (one set for each sloped segment), and two cutoff points.

1.3. For each input frame
1.4. For each detector
  1.4.1. Compare input Q with cutoff values and select appropriate quadratic coefficients.

  The input Q value is compared to the cutoff thresholds in the CPF, to determine which segment of the linearization to use.
  If input is less than cutoff #1, then segment 1 is used.
  If input is greater than or equal to cutoff #1 but less than cutoff #2, then segment 2 is used.
  If input is greater than or equal to cutoff #2, then segment 3 is used.

  1.4.2. Evaluate the quadratic function at the input Q:

  $output = qp[0, s] + qp[1, s] * input + qp[2, s] * (input)^2$

  where input = the input value, Q, in DN.
  output = the output value, Q_1, in linearized DN.
  qp[x, s] = linearization parameter x, for segment s.
  These parameters come from the CPF.

1.4.3. Return the output value.

7.6.5.5 Maturity

This algorithm will likely be correct for the flight FPA. A native C implementation will be completed by NASA by August, 2010 as part of the TIRS Science Data Processor pipeline, and the plan is to integrate it with the CalVal toolkit.

A process to analyze the on-orbit integration time sweep data and produce linearization Calibration Parameters will be discussed and produced at some later time.

Expect to have a more extensive test data set as other algorithms are completed.

More finely resolved test data for linearization will be generated for the flight FPA (not including the FPE), sometime in July, 2010 as the assembly level FPA tests are conducted.

Other possible changes that may occur are:

- **Lookup Table** – The remapping could also be implemented as a look-up table (LUT), which will return the "linearized" value (Q_1) for each instrument Q value. This table will be different for each detector and will return a floating-point number. The LUT is populated from the re-mapping function for each detector. This can be pre-calculated and stored as part of the calibration, or can be calculated at run time. This method may be quicker than calculating the re-mapping function as the curve fit functions and cut-off points don't have to be evaluated for each sample. It also allows a different function (other than quadratic) to be used to fit the linearization data. The LUT could also be generated with manual adjustments and delivered at part of a calibration input if necessary. A drawback is that a large floating point LUT (up to 4096 elements) for each detector must be held in memory during processing.

- **True Inconsistency Handling** – The inconsistency handling in the prototype code is included only because the initial test data (v3.0) is noisy and incomplete. It should not be necessary in LDCM final implementation – the TIRS linearization parameters should be well-understood and without errors by launch. If this turns out to not be the case, inconsistency handling may need to be included in the final code. It will resemble the current handling but should be more robust and comprehensive.

7.6.6 TIRS 40 Minutes Radiometric Stability Characterization

7.6.6.1 Background/Introduction

This algorithm provides for on-orbit characterization of the radiometric stability of the TIRS, specifically related to TIRS RD requirement (TIRS-547) "Thermal band data for all pixels, after radiometric calibration per 5.3.1.2, for radiometrically constant targets with radiances greater than or equal to the radiance corresponding to TTypical, shall not vary by more than plus or minus 0.7% (1-sigma) of their radiance over a 40 minute period." This algorithm is to be implemented, at least initially, as part of the Cal Val Tool Kit (CVTK).

For the TIRS apparent response stability is related to multiple contributing factors:
1) Background response and dark response stabilities over the 40 minutes

2) The reference constant source stability stabilities over the 40 minutes
3) Validity of CPF used to produce the radiometric product such parameters include the non-linearity correction and the conversion to radiance parameters.

In other words if any of these CPF parameters may have some thermal or temporal dependence that are not accounted for it would impact the interpretation of the actual response stability results.

Furthermore, although primarily focused on product stability this tool can also be used to understand the stabilities of several of the instrument sub-systems. Establishing baseline characteristics of the instrument stability will help in verifying the time it will take to return to nominal operation after lunar collects or other non-routine orbital operations. Baseline time duration for such return-to-nominal period would be based on telemetry information and could be complimented by the metric generated by this tool.

Key telemetry information associated with TIRS stability includes: Temperatures, (OBC, Telescope, SSM, FPA, FPE, MEB); Current and Voltages (OBC and FPE).

This algorithm uses deep space and OBC TIRS data to assess the within orbit product and instrument stability.
The basic concept of this algorithm is to analyze TIRS statistics from multiple 60sec segments within the stability evaluation interval of 36 min or 1.5 orbit. For each of the 60sec segments it compute the instantaneous stability parameters, which can be used then to gain understating on stability changes over the entire evaluation interval. Computing the 60sec stability parameters the algorithm is based on Hist_stat processing outputs.

Information about Input data types expected

Two types of data collects will be used for these analyses. on board calibrator data and deep space data.. The OBC data will be at or above the temperature of 290K (current plan is to use 290K, 295K and 315K) then, manual interpolation analysis will be required to transform the stability results into the requirement temperature of $T_{typical}$. Currently during on-orbit commissioning the main data collects that will be used by this algorithm are the continuous 36min data of OBC or deep space data as well as a repeated sequence of 30 Cal-sequence files over 1.5 orbit with once every 5 minutes collect sequence of 1 min deep space and 1 min OBC will be available.

This algorithm is designed primarily to work on TIRS data but could be converted to be applicable for both TIRS and OLI 36 minutes long collect. This algorithm can operate on all 3 TIRS bands (10µm,12µm, blind) but practical use will need only the 10µm and 12 µm bands data for product level performance assessment. For that reason the analyst can use a selection switch to select if this algorithm should process stability of TIRS blind data or just the imaging bands.
Similar to the 60sec stability algorithm this algorithm would rely on Histogram statistics Characterization processing to collect the basic statistics from either multiple scenes given in a sequence to form a virtual long interval or sub-segments of a long interval (that may be split into multiple files). The algorithm will populate the minimal set of radiometric stability metric as a standalone algorithm that is gathering the instantaneous stability information on each of the scenes or segments in the interval. In the case the scene provided is one long 36 min dataset (composed of several mission files) the algorithm will pre-process the input data to produce multi-segments statistics information needed for processing using a moving window of 60sec. The interpretation of the results requires that the scenes are temporally constant radiance, e.g., deep space and OBC

acquisitions (those could be part of a special Cal sequence sweep with enough dwell time for the collections of the number of lines needed).
For deep space collect we use (L0r) input that will only be non-linearity corrected. For OBC collects we use both (L0rc) and (L1r) products as inputs.

Information about Output produced and its interpretation

The output of this tool enables trending of : instantaneous stability of segments for the response, and stability over the 36min duration or longer.

The output mimics the 60s stability algorithm configuration where the output depends on the input data provided, i.e. OBC or deep space. Stability statistics across the scene segments are calculated from these uniform radiance scenes and stored in the characterization database. Using collects made at the nominal integration time the statistics are calculated by Histogram Statistics. Multiple levels of processing will be used by this algorithm. Six categories of outputs will be produced:

6) Background response band average stability (deep space, non-linearity corrected)
7) OBC response band average stability –(background corrected and non-linearity corrected)
8) Radiometrically corrected OBC response stability in radiance
9) Radiometrically corrected OBC response stability in %
10) Optional – Detector-by-detector Radiometrically corrected OBC response stability in % (for operability characterization)
11) Optional –Detector by detector background response stability (for operability characterization)

The OBC L1r product stability metric will be converted from radiance units to % so it can be evaluated against the 0.7% requirement level. OBC L0rc data will be used to characterize the band mean net response stability in the various segments over the data duration interval (36 min or longer).

Additional considerations

It is recommended stability acquisitions be routinely collected and assessed, in the following conditions: after standard TIRS operations in either earth sun-lit or night part of the orbit, after Lunar or Solar collects with 5 minutes margin from the time TIRS returned to earth view nadir pointing (the 5min period is driven by TIRS requirements and SC pointing stability allocated period).

The output of this algorithm can be used by subsequent algorithms that trend detector level characteristics.
TIRS radiometric conversion parameters are assumed invariant after the background corrected response is computed. Trending of the output of this stability algorithm could assist in validating this assumption.

Future developments on this characterization processing may include the use of OLI data and TIRS Earth - night ocean long passes.

To obtain a better representation of the TIRS stability will require additional analysis and correction of any influences of low frequency drifts in the TIRS and OBC temperatures induced by the data collection conditions. This will require the analysis to correlate between the trended results to the various subsystems telemetry information.

7.6.6.2 Inputs

This algorithm works with two types of source data a single long continuous collect or multiple short segments over a longer period..
The expected input will be produced by Hist_stat or the special long data Hist_stat algorithms which would operate on either a pre-processed data given in the format of a set of 30 4200 lines files (covering the period of 150 minutes) illustrating 30 interval sub-segments, or pre-processed data given in the format of a set of 36 4200 lines files illustrating 36 interval sub-segments (covering the continuous 36 min collect).
Inputs per segment are highlighted; inputs common to the full dataset are not highlighted. Note that when correlating telemetry data to this analysis it should be done in the same timing intervals of the segments used for the instrument response.

Descriptions	Symbol	Units	Level	Source
Interval Segment Signal Mean	$\overline{Q}_{B,S,D}$	Float [DN] or [w/m2 sr um]	N_BxN_SxN_D	Long collect statistic Characterization [15] Stat Char $\overline{Q}$
Interval Segment Signal Max	$Max\_Q_{(B,S,D)}$	Float [DN] or [w/m2 sr um]	N_BxN_SxN_D	Long collect statistic Characterization [1] Stat Char Q_{max}
Interval Segment Signal Min	$Min\_Q_{(B,S,D)}$	Float [DN] or [w/m2 sr um]	N_BxN_SxN_D	Long collect statistic Characterization [1] Stat Char Q_{min}
Interval Segment Signal StDev	$Sigma\_Q_{(B,S,D)}$	Float [DN] or [w/m2 sr um]	N_BxN_SxN_D	Long collect statistic Characterization [1] Stat Char σ
Number of frames	NBR_of_frames	Long		Long collect statistic Characterization [1] Stat Char *number of*

[15] Histogram data will be collected from different level of processing i.e. L0R for Dark Shutter data, L0Rc for OBC and at L1R for OBC product.

				frames
Interval segment ID or segment start line (line time tag)	Seg_ID	Float [s]	$N_{SEGMENTS}$	Long collect statistic Characterization Acquisition time for the first frame of each segment t_1
Scene Type (i.e O* or D*)	Stype	Char	N_B	DB L0Rp Image file
Processing Step (i.e. before or after Gain application)	Pstep	String	N_B	Histogram Stat Char Position in processing flow (RPS Level)
Segment window size (lines)	Seg_win	Int		Analyst input /pre-processing default set for 60sec duration – i.e. 4200 lines
Relative gains	r	[unitless]	N_BxN_SxN_D	CPF
Impulse Noise Pixels Locations	LM1	Integer	N_BxN_SxN_D xL	LM
Saturated Pixel Mask[16]	LM2	Integer	N_BxN_SxN_D	LM
Dropped Frames Mask	LM3	Integer	N_BxN_SxN_D xL	LM
Inoperable Detector List	Dinop	Float	N_BxN_S	CPF

7.6.6.3 Outputs

Outputs will mimic the provided segmentation of the input data (i.e. even if actual data was given as a single continuous collect since its inputs from the special long hist_stat processing will be chopped into multiple segments then the output DB will include results per segment – only in this case the segments are artificially produced by a moving window with in one or more mission data files).
Per interval segment populate these outputs
The algorithm will need to process the data from all segments before the analyst could assess the stability results for the full interval.
For Segment I the output to DB is :

Descriptions	**Symbol**	**Units**	**Level**	**Target**
Signal Variability, SCA average	$\overline{\Delta Q_{B,S}}$	Float [DN] or [w/m2	N_B x N_S	DB (Bias, Gains &

[16] This mask should include any of the detectors that had been reported by the Saturation characteristics processing, i.e. high and low saturations in both digital and analog categories

		sr um]		OBC L1R Stability)
L1r Product Variability, SCA average	$\overline{\Delta_{B,S}}$	Float [%]	N_B x N_S	DB (Gains & OBC L1R Stability)
SNR Variability, per-detector [17]	$\overline{\Delta_{B,S,D}}$	Float [%]	N_B x N_SxN_D	DB (Gains & OBC L1R Stability)
Scene Type (i.e O* or D*)	Stype	Char	N_B	DB
Processing Step	Pstep	String	N_B	DB
Interval segment ID or segment start line (line time tag)	Seg_ID	Float	$N_{SEGMENTS}$	DB
Segment window size (lines)	Seg_win	Int		DB

7.6.6.4 Options

Trending On/Off Switch: If trending is Off, output parameters are written to a text file.
Comp stat switch: ON – compute statistics, Off – import stat data from hist_stat (default)
Collect type ID – Cont 36min / 1.5 orbit data
Window sizes – full (default) , 4200, User selected between 100-64000 ; if collect type ID is Cont 36min the window size parameter will be used to define the segment size ; If Comp-stat switch is Off segment window size will be imported from hist_stat.
Processed Blind_band – Yes / No

7.6.6.5 Procedure

Note that in order to characterize TIRS radiometric stability we call this algorithm at least 3 times: Once to process long Deepspace data, and twice to process the data for the closest long OBC collects (once using the net linearized response L0rc1 intermediate Cal product and once using the radiometric corrected L1R product.)

Pre-Processing (creating sub segments generating the input data needed and computing Hist_stat info for each segment) The preprocessing brings either Collect type into a fixed input format.

1- Pre Process the data (if in multi mission data files continuous or multi-segments of Cal-sequence)

[17] We will not store in DB the OBC L1R calculated per detector gains 60 stabilities but it will be calculated and written to an output file if the option of trending is turned off.

Check if Collect type option is Cont 36min data

If No – get sequence of 30 scenes IDs to processes through this ADD

If yes chop the 36 min data into smaller segments in the size of Window_size input parameter

i.e. $Q_{interval}(b,s,d,l)$ - > { $Q_1(b,s,d,l)$; $Q_2(b,s,d,l)$; …. $Q_n(b,s,d,l)$ }

where the last line in Q_n is the last line in $Q_{interval}$ and all segment are set to equal size (Seg_win).

For a default window size there should be 36 scene IDs generated.

For all segments produce or obtain from DB the input histogram information needed for the algorithm.

2- On each scene or segment (30 or 36 segments) within the interval proceed to the following steps of 3-8

Calculating statistics (possibly can be retrieved from Hist Stat DB for the scene or segment)

3- Based on the Labeled Mask and detector operability list omit those detectors when calculating SCA level averages.

4- Populate histogram statistics metric per detector in the relevant variables and calculate the average across all detectors within an SCA.

a. $\overline{Q}_{(B,S,D)} = Histogram\_per\_detector\_mean\_signal\ \overline{Q}$ (1)

b. $Max\_\overline{Q}_{(B,S,D)} = Histogram\_per\_detector\_Max\_signal\ Q_{max}$ (2)

c. $Min\_\overline{Q}_{(B,S,D)} = Histogram\_per\_detector\_Min\_signal\ Q_{min}$ (3)

d. $Sigma\_\overline{Q}_{(B,S,D)} = Histogram\_per\_detector\_StDev\ \sigma$ (4)

e. $\overline{Q}_{B,S} = mean(\overline{Q}_{B,S,D})$ (5)

f. Get *number_of_frames* for the interval segment

g. Get Interval segment ID from input (this will be related to the segment's first line time tag)

h. Note that calculations should only include operable and in-spec detectors. (i.e. ignore pixels flagged as inoperable, saturated, dropped frame, impulse or fill.)

Calculating Stability of Signal

5- For each detector, calculate the 1-sigma the along track variation in the image that is at the length of the segment window or segment duration.

$$\Delta Q_{(B,S,D)} = 1 \times Sigma\_\bar{Q}_{(B,S,D)} \quad (6)$$

Processing data to generate outputs

6- Calculate the SCA average variability, i.e, the average across all detectors within each SCA:

b. $\overline{\Delta Q_{B,S}} = mean(\overline{\Delta Q_{B,S,D}})$ (7)

7- For TIRS deep space data, record the SCA mean variabilities ($\overline{\Delta Q_{B,S}}$) of every segment or scene within the interval to the database or output file (along with other specified outputs in the output table). This is the end of the algorithm for processing TIRS deep space data.

8- For TIRS OBC data process and record what is done up to step 7 and also do this :

f. Calculate the per-detector variability in terms of percent change. It illustrates the impact of individual detectors on the overall radiometric stability.

iii. $\overline{\Delta_{B,S,D}} = r_D * \frac{\overline{\Delta Q_{B,S,D}}}{\overline{Q}_{B,S,D}} * 100\%$ (8)

iv. where r_D is the relative linear gain used for conversion to at aperture radiance from CPF for L0Rc data and 1.0 for L1R data.

g. Calculate the SCA average variability in terms of percent gain change:

iii. $\overline{\Delta_{B,S}} = \frac{\overline{\Delta Q_{B,S}}}{\overline{Q}_{B,S}} * 100\%$ (9)

iv. $\overline{Q}_{B,S}$ should be calculated using the same list of detectors used to produce the mean signal in equation (5)

h. For L0Rc data, write the per-detector percent variability ($\overline{\Delta_{B,S,D}}$) and the SCA average variability ($\overline{\Delta_{B,S}}$) of every segment or scene within the interval to the database or output file (along with other specified outputs in the output table).

i. For L1R data, write the SCA average percent variability ($\overline{\Delta_{B,S}}$) of every segment or scene within the interval to the database or output file (along with other specified outputs in the output table).

j. For L1R if output to file selected also write to file the per-detector variability ($\overline{\Delta_{B,S,D}}$) and the SCA Radiance mean variability ($\overline{\Delta Q_{B,S}}$)of every segment or scene within the interval (along with other specified outputs in the output table).

9- Once all segments or scenes had been processed analyst will work offline to extract information from DB to generate various plots and test correlations to telemetry data. Most of this follow-on analysis is not automated or strictly defined by this algorithm but it may ultimately include plots of the SNR variability over the interval and use such plots to identify detectors are 5 time less stable than the overall band stability characteristics assisting with detector operability characterization. Other likely plots are plots similar to lamp response trending – trending mean response Vs. time and stdev of response Vs. time in each segment over the 36 minute or longer interval.

7.6.6.6 Maturity

Level 2 (reuse portion of 60s stability algorithm).
The algorithm essentially repeats the 60s stability computations (with slight modification) per segment within the long interval being evaluated. The only part that is new is that it is capable of working with a sequence of individual small scenes or one long scene that get spliced into a moving window segments. This ADD enable processing that can either work on the data directly – or to be a stand-alone algorithm that uses the DB produced by a special variant of the histogram statistics algorithm which computes the information per segment for a long continuous collect. The output table should be linked to an single interval ID , and the output metric is stored per segment will be linked to its corresponding interval. Regardless of the version of the data input provided (a one long file or multiple segment files) the output in DB should be organized the same way.
Change from 60s Stability - 2x Sigma in Eq. (6) is now only 1 x Sigma

7.6.7 TIRS Background Response Determination

7.6.7.1 Background

The significant components of the signal recorded by each TIRS detector are the Dark Response (from thermal energy of the focal plane), the Background Response (photon energy from the instrument structure and optics), and the Scene Response (photon energy from the optical target of the detector). These input stimuli combine to produce an electrical potential across a capacitor that is read out by the ROIC electronics. The conversion efficiency of the input energy to the electrical potential varies for each detector. Each capacitor's potential (considered a count of electrons in a detector well) is digitized by the ROIC A/D to a quantized signal (Q) in units of digital number (DN) though a transistor network that also varies for each detector. Since these two conversions are difficult to separate in experimentation, and not possible in-flight, the totals system conversion of input (energy) to output (Q) is considered as a unit whenever possible.

A key target used to determine the background response of the TIRS instrument is deep space where there is negligible scene response. The dark response is present in all measurements. Since the focal plane temperature is carefully controlled, the dark response is assumed to be constant. A correction for variation of the dark response on a scene-by-scene basis is calculated in another algorithm (Dark Response Determination).

This algorithm analyzes the data taken while looking at space to determine the background response needed to remove the per-detector "bias", which is the sum of the background and dark response. The baseline dark response, which is provided in the CPF, is subtracted from the bias to provide the background response.

While this algorithm is designed to derive only the background response, the linearity function (or lookup table), background response, and gain should be looked upon as a hierarchy in the listed order, each set dependent on accurate values of the previous set.

This algorithm calculates the output of the system in the normal science mode configuration (integration time), with negligible scene response. In a perfectly linear system, this could be used as a simple offset from which other measurements are calibrated. In a non-linear system, this determines where on the linearization curve the zero response is placed. Linearization can then be

applied to the Scene data using the differential from this bias point, or applied to data from the combined input response allowing the bias level to be directly subtracted.

7.6.7.2 Input

Description	Symbol	Units	Level	Source	Type
Scene Mean	$\overline{Q}$	DN	N_{scenes} x N_{band} x N_{SCA} x N_{det}	DB (Histogram Statistics)	Float
Baseline Dark Response	D_B	DN	N_{band} x N_{SCA} x N_{det}	CPF	Float

7.6.7.3 Output

Description	Symbol	Units	Level	Target	Type
Background Response	B	DN	N_{band} x N_{SCA} x N_{det}	CPF	Float
Background Response Standard Deviation	σ_B	DN	N_{band} x N_{SCA} x N_{det}	Report	Float

7.6.7.4 Options

- List of space-look scenes for analysis
- Optional report

7.6.7.5 Procedure

1. Read the scene means for each space-look scene selected for analysis.
2. Read the baseline dark response from the CPF.
3. Subtract the baseline dark response from each scene mean.
4. Determine the mean and standard deviation across all selected scenes' mean of the background response for each band/SCA/detector. The standard deviation allows the CVT to evaluate the Background Response. The mean is the value to be placed in the CPF, if further analysis shows an improvement in product quality.

7.6.7.6 Maturity

Level 3 Further component and higher level testing of the TIRS QWIP detectors is necessary before the details of this algorithm are resolved.

7.6.8 TIRS Gain Determination

7.6.8.1 Background

The significant components of the signal recorded by each TIRS detector are the Dark Response (from thermal energy of the focal plane), the Background Response (photon energy from the instrument structure and optics), and the Scene Response (photon energy from the optical target of the detector). These input stimuli combine to produce an electrical potential across a capacitor that is read out by the readout integrated circuit (ROIC) electronics. The conversion efficiency of the input energy to the electrical potential is different for each detector. Each capacitor's potential (considered a count of electrons in a detector well) is digitized by the ROIC A/D to a quantized signal (Q) in units of digital number (DN) through a transistor network that also varies for each detector. Since these two conversions are difficult to separate by pre-launch testing and not possible in-flight, the total system conversion of input (energy) to output (Q) is considered as a unit whenever possible.

The calibration strategy of the TIRS instrument is to provide measurements with negligible scene response (looking at space), and measurements with known scene response (looking at the on-board calibrator). This determines a measurement scale to be applied to the scene response detected when viewing the target (Earth). The dark response is present in all measurements. Since the focal plane temperature is carefully controlled, the dark response should be constant. A correction for variation of the dark response on a scene-by-scene basis is calculated in another algorithm (Dark Response Determination).

This algorithm processes the data taken viewing the on-board calibrator, which has already been corrected for dark and background response determined in part by viewing deep space, to determine the calibration gain function to apply to Earth scene data. By design, the on-board calibrator will have a known spectral radiance, modeled as a black-body corrected for an emissivity near unity, and will illuminate each detector through the entire optical system,. The spectral radiance can be varied by changing the calibrator temperature set point. The TIRS detector response to the black-body data must be corrected for non-linearity as well as the dark and background response.

The relationship between linearized, background subtracted DNs and radiance behaves fairly linearly within instrument's designed range of operation. Therefore, a linear gain function including a slope and an offset is determined to convert linearized, background subtracted DNs to radiance.

7.6.8.2 Input

Description	Symbol	Units	Level	Source	Type
Black-Body Scene Mean (non-linearity corrected)	$\bar{Q}$	DN	N_{scenes} x N_{band} x N_{SCA} x $N_{detectors}$	DB (Histogram Statistics)	Float
Weighted Average Black Body Spectral Radiance Lookup Table	$L_{\lambda,lookup}$	$W/m^2 sr \mu m$	N_{band} xN_{SCA}xN_{det} $N_{radiances}$	File*	Integer

Pre-acquisition Deep Space Averages (non-linearity corrected)	S_a	DN	$N_{scenes}xN_{band}$ x N_{SCA} x N_{det}	BPF	Float
Post-acquisition Deep Space Averages (non-linearity corrected)	S_b	DN	$N_{scenes}xN_{band}$ x N_{SCA} x N_{det}	BPF	Float
Dark Response (non-linearity corrected)	D	DN	$N_{CPF}xN_{band}$ x N_{SCA} x $N_{detectors}$	CPF	Float
Background Response (non-linearity corrected)	B	DN	$N_{CPF}xN_{band}$ x N_{SCA} x $N_{detectors}$	CPF	Float
Black-Body Temperature	T_{BB}	K	$4^{\dagger}xN_{scenes}xN_{samples/scene}$	Ancillary data/DB	Float
Black Body Temperature Weighting Factors	W_{BB}	Unitless	$4^{\dagger}$	CPF	Float

*** $L_{\lambda,lookup}$ are calibration parameters that won't change once determined prior to launch.**
†The '4' here is from the number of thermistors on the black body

7.6.8.3 Output

Description	Symbol	Units	Level	Target	Type
Absolute Gains	G_{abs}	$\frac{DN}{W/m^2 sr \mu m}$	N_{band} x N_{SCA}	Test CPF/Report	Float
Relative Gains	G_{rel}	Unitless	N_{band} x $N_{SCA}xN_{detectors}$	Test CPF/Report	Float
Gains	G_S	$\frac{DN}{W/m^2 sr \mu m}$	N_{band} x $N_{SCA}xN_{detectors}$	Report	Float
Gain Offset	c_G	DN	N_{band} x $N_{SCA}xN_{detectors}$	Test CPF/Report	Float
Detector Gain Standard Deviation	σ_G	$\frac{DN}{W/m^2 sr \mu m}$	N_{band} x N_{SCA} x $N_{detectors}$	Report	Float

Date range of black-body scenes			1	Report	Text

7.6.8.4 Options

- Date range of black-body scenes for analysis
- Optional report
- Dark and Background response selection
    1. Pre-acquisition deep space averages (S_a)
    2. Post-acquisition deep space averages (S_b)
    3. Average of pre-and post-acquisition deep space averages (S_{ab}) (default)
    4. dark and background responses from the CPF (or dark response determination in the case of dark response)

7.6.8.5 Procedure

For each spectral band, SCA, detector, and black-body scene selected for analysis:

1. The combined dark and background response S_{BD} will need to be removed from all black body collects. First determine S_{BD} based on the selected option. If pre- or post-acquisition deep space averages have been selected, then retrieve the selected deep space averages and they will be S_{BD}.

   If the average of the pre-and post-acquisitions is selected then retrieve the selected deep space averages and calculate S_{BD} using equation (1)

   where:
   b is the spectral band
   s is the SCA
   d is the detector

$$S_{BD}(b,s,d) = \frac{1}{2}\left(S_A(b,s,d) + S_B(b,s,d)\right) \qquad (1)$$

   If the dark and background responses from the CPF are desired, then retrieve the dark and background responses from the CPF(s) (or dark response determination in the case of dark response) and calculate S_{BD} using equation

   (2).

$$S_{BD}(b,s,d) = D(b,s,d) + B(b,s,d) \qquad (2)$$

   Now that the combined dark and background response has been determined, remove it from the per detector means using equation (3) where S represents a black body scene. If the combined response is to have come from the CPF, make sure that each black body collect is corrected by a CPF that is valid for the time those data were collected. This means that multiple (N_{CPF}) CPFs may need to be used. If not then N_{CPF}=1. Note that all these data should have previously been linearized.

$$\overline{Q_{BS}}(S,b,s,d) = \overline{Q(S,b,s,d)} - S_{BD}(b,s,d) \qquad (3)$$

2. Calculate the average black body temperature for each scene. Black body temperature samples are collected once per second.

$$T_{BB,scene}(t,S) = \frac{1}{N_{Samples}} \sum_{1}^{N_{samples}} T_{BB}(t,S,sample) \quad (4)$$

where t is an index that identifies each thermistor and $N_{samples}$ is the number of samples.

3. Calculate the weighted average of the black body temperatures

$$T_{BB,avg}(S) = \frac{1}{\sum_{t=1}^{4} W_{BB}(t)} \sum_{t=1}^{4} W_{BB}(t) T_{BB,scene}(t,S) \quad (5)$$

4. Determine the average black body spectral radiances for each detector for each scene using $T_{BB,avg}$ and the black body spectral radiances lookup table.

$$L_{\lambda}(S,b,s,d) = L_{\lambda,lookup}(T_{BB,avg}(S),b,s,d) \quad (6)$$

5. Using a least squares fit, find the relationship (7) between $\overline{Q_{BS}}$ and the average black body spectral radiances (L_{λ}) for the corresponding black body scenes. Write the gain offsets c_G to the CPF.

$$L_{\lambda}(S,b,s,d) = \frac{\left[\overline{Q_{BS}}(S,b,s,d) - c_G(b,s,d)\right]}{G_s(b,s,d)} \quad (7)$$

6. Calculate the absolute gains for each SCA and band from the gain slopes and write them to the CPF.

$$G_{abs}(b,s) = \frac{1}{N_{\det}} \sum_{1}^{N_{\det}} G_S(b,s,d) \quad (8)$$

7. Calculate the relative gains for each detector using the gain slopes and the absolute gains and write them to the CPF.

$$G_{rel}(b,s,d) = \frac{G_s(b,s,d)}{G_{abs}(b,s)} \quad (9)$$

8. Calculate the standard deviation of the relative gains across the entire focal plane. Note here that N_{det} represents the total number of detectors across the entire focal plane.

$$\mu_G(b) = \frac{1}{N_{\det}} \sum_{N=1}^{N_{\det}} G_{rel}(b,d) \quad (10)$$

$$\sigma_G(b) = \sqrt{\frac{1}{N_{\det}} \sum_{1}^{N_{\det}} \left(G_{rel}(b,d) - \mu_G(b)\right)^2} \quad (11)$$

9. If an output report is chosen, write the gain function slope (G_S), the relative gains (G_{rel}), the absolute gains (G_{abs}), offset(c_G), and standard deviations (σ_G) to the output report.

7.6.8.6 Maturity

Depending how large the weighted average spectral radiance lookup table is, linear interpolation may not be needed. The trade-off is a larger lookup table or linear interpolation.

Potential modifications to this algorithm include:

1. Storing the per-scene detector gains in the characterization database.

2. The dark response determination algorithm may need to run prior to this algorithm if the baseline dark response isn't sufficient. This would add an option to this algorithm of which dark response should be used. It also implies that the dark response would be per-scene.

Section 8 Lexicon

Acronym	Definition
ACS	Attitude Control System
ASC	
ASCII	American Standard Code for Information Interchange
AVHRR	Advanced Very High Resolution Radar
B2B	Band-to-Band
BIH	Bureau International de l'Heure
CPF	Calibration Parameter File
DC	
DCE	Data Collection Event
DEM	Digital Elevation Model
DMA	Defense Mapping Agency
DOQ	Digital Orthophoto Quadrangles
ECF	Earth Centered Fixed
ECI	Earth Centered Inertial
ECR	Earth Centered Rotating
ECS	EOSDIS Core System
EF	Earth Fixed
EO-1	Earth Observing-1
EOSDIS	Earth Observing Station Data and Information System
ER	
EROS	Earth Resources Observation System
ETM	Enhanced Thematic Mapper
GCP	Ground Control Point
GCTP	General Cartographic Transformation Package
GPS	Global Positioning System
GSD	Ground Sample Distance
HDF	Hierarchical Data Format
IAS	Image Assessment System
IEEE	Institute of Electric and Electronic Engineers
IFOV	Instantaneous Field of view
IRU	
LAS	
LOS	Line-of-Sight
MD	Metrics Database
MET	Spacecraft Time
MIT	Massachusetts Institute of Technology
MLH	Maximum Likelihood Estimate
MS	Multispectral
NASA	National Aeronautics and Space Administration
NBR	Navigation Base Reference
NEOS	National Earth Orientation Service
OB	Orbit Reference Frame
OLI	Operational Land Imager
PAN	Panchromatic

RMSE	Root Mean Squared Error
SCA	Sensor Chip Assemblies
SDP	
SDS	
SOM	Space Oblique Mercator
TAI	International Atomic Time
TIRS	Thermal Infrared Sensor
USGS	U.S. Geological Survey
UT-1	Universal Time
UTC	Universal Time-Coordinated
UTCF	
UTM	Universal Transverse Mercator
WGS84	World Geodetic System 1984
WLS	Weighted Least Square
WRS	World-wide Reference System

Section 9 References

Reference 1
Bicknell, W. E., Digenis, C. J., Forman S. E., Lencioni D. E., “EO-1 Advanced Land Imager”, MIT Lincoln Laboratory.

Reference 2
Lencioni, D. E., Digenis, C. J., Bicknell, W. E., Hearn, D. R., Mendenhall, J. A., “Design and Performance of the EO-1 Advanced Land Imager”, MIT Lincoln Laboratory.

Reference 3
Landsat 7 Image Assessment System (IAS) Geometric Algorithm Theoretical Basis Document.

Reference 4
Bate, R. R., Mueller D. D., and White, J. E., “Fundamentals of Astrodynamics”, Dover Publications, 1971.

Reference 5
Brown, R.G., “Introduction to Random Signal Analysis and Kalman Filtering”, John Wiley and Sones, New York, 1983

Reference 6
Haykin, Simon, “Adaptive Filtering Theory”, Prentice Hall, 1991

Reference 7
Snyder, John P., Map Projections - A Working Manual, United States Geological Survey Professional Paper 1395, U. S. Government Printing Office, Washington, 1987.

Reference 8
Park, S.K., R.A. Schowengerdt, Image Reconstruction by Parametric Cubic Convolution, Computer Vision, Graphics and Image Processing, v.23, no.3, September 1983.

Reference 9
Rosborough, G.W., D.G. Baldwin and W.J. Emery, "Precise AVHRR Image Navigation", IEEE Transaction on Geoscience and Remote Sensing, Vol. 32, No. 3., May 1994.

Reference 10
Rao, C.R., "Linear Statistical Inference and Its Applications", John Wiley and Sons, Inc., 1973.

Reference 11
Sahin, M., P.A. Cross, and P.C. Sellers, "Variance Component Estimation Applied to Satellite Laser Ranging", Bulletin Geodesique, 66: 284-295, 1992.

Reference 12
Yuan, Dah-ning, "Optimum Determination of the Earth's Gravitational Field from Satellite Tracking and Surface Measurements", Dissertation, The University of Texas at Austin, 1991.

Reference 13

Snyder, John P., Space Oblique Mercator Projection Mathematical Development, United States Geological Survey Bulletin 1518, U. S. Government Printing Office, Washington, 1981.

Reference 14
LAS 6.0 Geometric Manipulation Package Overview Document, USGS EROS Data Center.

Reference 15
Cook, R. Dennis, “Detection of Influential Observations in Linear Regression”, Technometrics, Volume 19, Number 1, February, 1997.

Reference 16
Golub, Gene H. and Charles F. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, MD, 1983.

Reference 17
DMA TR 8350.2-A, DMA Technical Report, Supplement to Department of Defense World Geodetic System 1984 Technical Report, prepared by the Defense Mapping Agency WGS84 Development Committee, dated December 1, 1987.

Reference 18
Landsat 7 System Program Coordinates System Standard, Revision B, prepared by Martin Marietta Astro Space, document number PS23007610B, dated 2 December 1994.

Reference 19
Theoretical Basis of the Science Data Processing Toolkit Geolocation Package for the ECS Project, prepared by the EOSDIS Core System Project, document number 445-TP-002-002, dated May 1995.

Reference 20
IAS Radiometry Algorithm 6.6a, “Outliers in a Regression”, Goddard Space Flight Center, 1996.

Reference 21
Press W., Teukolsky S., Vetterling W., and Flannery B., Numerical Recipes in C, 2nd edition, Cambridge University Press, 1992.

Reference 22
Wolberg, G., Digital Image Warping, IEEE Computer Science Press, 1990.

Reference 23
Gonzalez, R., Wintz, P., Digital Image Processing, Addison-Wesley Publishing Company, 1987.

Reference 24
Akima, Hiroshi, “A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures”, Journal of the Association for Computing Machinery, Vol. 17, no. 4, October 1970.